**EPISODE 1048**

[INTRODUCTION]

**[00:00:00] JM:** Shopify is a platform for selling products and building a business. Shopify is a large e-commerce company with hundreds of engineers and several different mobile apps. Shopify's engineering culture is willing to adapt new technologies aggressively, trying new tools that might provide significant leverage to the organization, and React Native is one of those technologies. React Native can be used to make cross-platform mobile development easier by allowing code reuse between Android and iOS. React Native was developed within Facebook and has been adapted by several other prominent technology companies with varying degrees of success. Many companies have seen improvements with their mobile development and release process such as Discord and many other companies.

However, in a previous episode, we talked with Airbnb about their adaption of React Native. That adaption was less successful. Of course, that was a few years ago. Farhan Thawar is a VP of engineering at Shopify. He joins the show to talk about Shopify's experience using React Native, the benefits of cross-platform development and Farhan's perspective on when it is not a good idea to use React Native.

[SPONSOR MESSAGE]

**[00:01:20] JM:** Today's show is sponsored by Datadog, a scalable, full-stack monitoring platform. Datadog synthetic API tests help you detect and debug user-facing issues in critical endpoints and application. Build and deploy self-maintaining browser tests to simulate user journeys from global locations. If a test fails, get more context by inspecting a waterfall visualization or pivoting to related sources of data for troubleshooting. Plus, Datadog's browser tests automatically update to reflect changes in your UI so you can spend less time fixing tests and more time building features.

You can proactively monitor user experiences today with a free 14-day trial of Datadog and you will get a free t-shirt. Go to softwareengineeringdaily.com/datadog to get that free t-shirt and try out Datadog's monitoring solutions today.

 

[INTERVIEW]

**[00:02:28] JM:** Farhan, welcome to Software Engineering Daily.

**[00:02:31] FT:** Thanks for having me.

**[00:02:32] JM:** You're the VP of engineering at Shopify. You joined the company a year ago. What have you learned about how Shopify builds software that surprised you?

**[00:02:42] FT:** Well, this is a very loaded question, because two reasons it's loaded. One is that Shopify does many things differently, and I have been around the block enough to see how many companies approach software, and Shopify is definitely different. I would say it's loaded also because when I started at Shopify, it was the number one question people asked me. People kept asking me when I joined, "What's the most surprising thing about Shopify?" I think, internally, people know or feel like it is a different type of engineering environment.

I'll start off with what I think is different. In many companies I've worked at, what the company tries to do is really focus on efficiency. When you focus on efficiency, you really try to make teams as distributed and disaggregated as possible, because you're really trying to focus on how do I make the team move quickly without running into another team.

What I found at Shopify is that this view, while can work in many instances, really leads you towards like a local maxima. If you know exactly what you want to do and what you want to build, you can try to build the efficient team structure that'll get you there. The Shopify view is we don't want to land on a local maxima. We want to build the right thing for merchants. What that means is introducing a little more ambiguity. Introducing a little more friction so that you are rubbing up against other teams, and
with that type of model, we believe we'll end up with an overall better, long-term solution for our merchants.

**[00:04:23] JM:** Most of the listeners are probably aware of what Shopify is. They probably used it. How do users interact with Shopify on mobile devices?

**[00:04:35] FT:** There's two or three different audiences on mobile to interact with Shopify. The one that most folks will know about is basically as the buyer. If you're a buyer, and Shopify has hundreds and millions of buyers, because we power all sorts of stores all over the Internet, you will interact with it via mobile web. You'll be browsing for products. You'll interact with an e-commerce store and then you'll do an entire purchasing flow over mobile web, and that's how you'll end up interacting with Shopify. We make that as pleasant and as seamless as possible, and there's all sorts of amazing technology behind that to make it as fast and as delightful as we want our buyers to feel. That's first one way.

Another way to interact to Shopify over mobile is actually as a merchant. As no surprise to any of your listeners, the mobile ecosystem over the last, really, 13 years since the iPhone came out, has revolutionized how we operate. We don't feel like we have to be tied to our desks in order to do our jobs. Many, many merchants interact with Shopify via Shopify Mobile, which is our mobile apps, and that's a way to really feel like you can manage your entire storefront, your entire store from your phone or your tablet.

Then the third way is as a retailer. We're in an interesting time period right now with COVID-19, but as a retailer who has a retail storefront, we have our point-of-sale app. The point-of-sale app is really tailored to enable you to transact with your physical retail environment, store operations, reporting, inventors, all those things will be managed. Again, that's a mobile application that you'd be interacting with as someone who works in that retail store.

**[00:06:27] JM:** As I understand, Shopify also has an API that is used quite widely. Do people build their own mobile apps on top of that API? Do people build mobile apps that use Shopify as an ecommerce backend API situation?

**[00:06:49] FT:** Yeah. You're hitting on one of the not so secret secrets of Shopify, and that we build things platform-first. What that means is we try to enable our engineering team to build upon the same APIs that we expect all of our partners to build on top of, and we do that because it allows everyone to be building on top of an amazing framework to enable things for merchants.

Now, on the ecommerce side is it's very similar to the mobile side and that you can build mobile experiences through the same APIs. Actually, it's enabled in a few different ways. One is, is that you can enable a buyer experience, and there are partners who do this all the time, where they build an entire mobile app for themselves. Interact with the Shopify APIs, the Shopify platform, and build a really great buying experience that you may not be able to get from just using the mobile web, and they want it to be completely custom. They want to potentially enable notifications and other types of interactions so they can the entire mobile experience, mobile app experience.

There are partners that we work with as well that can help you build a mobile app experience. Then you can actually build things directly on top of our platform right into our POS as well. There are many examples of partners building things that enable like loyalty, or recommendations, or client-telling right through our Shopify point-of-sale application. So you can build something that lives as part of an adjunct to the checkout, for example. You may be in a retail store. You're going to use the point-of-sale application to check out your customers. During that flow, you're able to add in loyalty numbers from buyers. You're able to have client-telling apps. So as a retail staff, you might have some downtime, in which case you want to message some of your VIPs and let them know about what's in your inventory or what could match with something about last week. All those applications can live right inside the POS app as well, and they look and feel just like Shopify POS. Point-of-sale when I say POS.

**[00:08:59] JM:** Okay. Talking about Shopify's engineering culture, Shopify engineering tries to adapt new technologies pretty aggressively. That will include React Native, as we'll talk about. On the other hand, Shopify is core infrastructure for its customers. How do you balance the adaption of new technologies while also maintaining high reliability?

**[00:09:31] FT:** Yeah, it' a good question. It is part of the extension to my answer at the beginning, and that Shopify does have quite a different engineering culture than most other companies. The way I can explain it is the following. We really do care about new technology. We want to make sure that we use technology as leverage, because that's what it is. However, we only want to adapt new technology if we feel like we're getting a step change in the leverage that we get from the technology.

It could be that the developer experience is much better. It could be were' going to get leverage on performance. It could be ready to get leverage on the types of people we can hire. So there're all sorts of reasons why we want to get leverage, and we'll only adapt the technology if it satisfies those conditions. We're not looking to pick at will any new technology off-the-shelf just because it's new. We're not trying to do that, but we are trying to make sure that we have an opinion about the technology we choose.

I would say the second thing we want to do is when we do choose something, we want to go all-in. So we're not taking something off-the-shelf and saying, "Okay, well let's use it for this little side project in this one instance." What we're trying to say is, "Can this be a core primitive for Shopify engineering?"

A good example is Ruby and Ruby on Rails. It's not the default choice for all applications in the world as you know, but it is something that we've adopted quite early on. When we adapt something, we become core contributors. We push back much to the open source community. We're able to have opinions about how it should work. In the case of Ruby on Rails, we do really feel that it was a lever. It made Shopify able to move very pretty quickly in the early days and even in the late stages we are in now. As you know, Shopify is still a big Ruby and Ruby on Rails shop. That's the type of leverage that we want to get from technology.

But in order for things to meet that bar, it goes through a process by which we can understand where we think we're going to get that leverage, and then we do adapt it, like I mentioned, we want to become core to the community surrounding that technology. It's not something that people just say, "Oh, yeah. Shopify also uses it." It's very clear that we use it, because we're in the community in a big way.

**[00:11:57] JM:** Let's move towards a discussion of React Native. Can you give me an overview of Shopify's mobile tooling infrastructure before the company started looking seriously at React Native?

**[00:12:12] FT:** Sure. Like I mentioned earlier that in '07 when the iPhone came out, '08, was when the app store came out. This is the time that a lot of folks started realizing that smartphones would be everywhere. The early days of application development, Native was

really the only choice you have. Native development on iOS and Android was really the only clear choice in which you can build great mobile experiences.

Now, there were other ways, but it was pretty normal for folks to try some of the alternative approaches, right? PhoneGap and Sentia. My background is actually from a mobile agency called ExtremeLabs, and we had tested and tried many of these cross-platform frameworks including HTML 5. But at the time, the only thing that really gave you that high-performance fluid UI experience was native development. It's not surprising that that's where Shopify started also in building mobile applications.

Now, what's amazing about native application development is you're much closer to the metal. You have great tooling infrastructures from Google and Apple in this case. You are able to move along that tool chain, easier debugging. Before React Native came along, that was 100% of Shopify's mobile infrastructure.

[SPONSOR MESSAGE]

**[00:13:43] JM:** If you can avoid it, you don't want to manage a database, and that's why MongoDB made MongoDB Atlas, a global cloud database service that runs on AWS, GCP and Azure. You can deploy a fully-managed MongoDB database in minutes with just a few clicks or API calls and MongoDB Atlas automates deployment and updates and scaling and more so that you can focus on your application instead of taking care of your database. You can get started for free at mongodb.com/atlas, and if you're already managing a MongoDB deployment, Atlas has a live migration service so that you can migrate your database easily and with minimal downtime, then get back to what matters. Stop managing your database and start using MongoDB Atlas. Go to mongodb.com/atlas.

[INTERVIEW CONTINUED]

**[00:14:42] JM:** React Native originally came out in 2015, and Shopify looked at it for a while, but it was not sufficient back then. What were the shortcomings of React Native back in 2015 when the project first came out?

**[00:14:58] FT:** Yeah. You're right. It came out at Facebook in 2015, and that's actually the time that Shopify did their first deep dive. I think what's true in all of these mobile frameworks is that whatever you want to build is likely possible, but it might not be effective or efficient for you to build those apps in that way. What I mean by that is there are tried-and-true ways in which you want to spend your time building mobile apps. If you're able to build something effectively in that platform and the output to the customer is actually compelling, you'll use that framework.

In native development, it was the case that you could actually build native applications. It wasn't too onerous. Like I mentioned before, the debugging infrastructure was there, the tool chain was there and people felt like, “Okay, this is a good use of engineering time and I’m getting the appropriate output from native development such that my buyers in the Shopify case or my merchants would like this experience.”

React Native in 2015 was not like that. It was very onerous to build things. It did not have great Android support. One of the reasons to use a tool chain like React Native is to be able to get both platforms by building, in on theory, they call it learn once, run anywhere. So you're mostly building on one platform. Maybe there is a small percentage of code that is unique to each platform, but otherwise it's shared code. This was not the case in 2015. There was a combination of performance problems, tool chain problems. A lot of things were actually still internal to Facebook at the time, so not fully open source, of course, Android support. The line that we used back then to describe our testing efforts was we would not be able to build an app that we would be proud of using React Native at this time.

**[00:16:50] JM:** What changed in that period of time since 2015? How did React Native get up to snuff such that the quality was good enough to build Shopify's apps?

**[00:17:05] FT:** Yeah. A few things changed. I mean, one was that much more support came to the ecosystem. Actually, may be surprising to the listeners is that much of the infrastructure for React Native today came in the last year or so. So there's been a reinvestment from Facebook over the last year and a bit into React Native. Actually, maybe even going back two years, that really spearheaded the latest set of people jumping onto the framework.

React Native came out. There was definitely some companies adapting it. Some adapting it and moving off of it. You've got a bunch of podcasts about Airbnb and React Native. I would say, really, around 2018 and 2019 was when a lot of the push happened to get it up to where it was today.

Now, what changed specifically? I think one was really first-class Android support. Two was a real focus on performance. Three was other companies contravening back to React Native, such that it's not just Facebook. It's actually now you see Microsoft is there with much of the infrastructure help. Shopify is there now helping with and make it a real community project. So it's not just Facebook. Then you're starting to see really compelling applications being built that are really highly-rated and also very popular come out in the app store.

I think this was not necessarily true in the early days of React native. There was lots of comments around like, "Well, show me some great apps," and then people would show you great apps and they would say, "Oh, yeah. But show me like a popular app or shoe me a top 10 app." That wasn't the case then, but is the case now.

**[00:18:41] JM:** The benefits of React Native, I think it's worth exploring these in the context of Shopify, because there are a lot of ways that React Native can provide a benefit. It's in JavaScript. So you could theoretically pull in dependencies on the fly over the web. It's kind of cross-platform. You can theoretically port some code from one place to another. Tell me the benefits of using React Native in terms of Shopify.

**[00:19:15] FT:** Yeah. I think there're a few reasons why it was the right choice for Shopify and maybe the right choice for other companies. One of the reasons that people want to look at a framework like React Native is definitely for cross-platform development. All of these mobile companies that build mobile apps and want to have a mobile applications in the app store, they always try to look to one of these frameworks because they're trying to cut their development time in half.

I would say React Native was one of the first ones to come out, which actually is now starting to fulfill that promise. You mentioned everything written in JavaScript. What that means for Shopify and many other companies is that you potentially can have your engineers who work on your

desktop and mobile web contribute to your mobile projects. I think that's a really great example of having that learn once, run anywhere ecosystem.

If you're writing React on the web, you can, with some small tweaks, because React and React Native are not the same, with some small tweaks actually be able to contribute to a mobile project that ends up being a native implementation on React Native. I think that's important.

I think two, one of the things that we found at Shopify was while we thought we'd get some code share between iOS and Android, I didn't think we were expecting the level of co-chair that we got. I've been around a while and I was expecting at a high-end 70% or 80% code share, which is amazing when you're talking about mobile that you could only have 20% to 30% of a platform-specific code. But we're seeing 95%, and in some cases 99% code sharing, which it's a much higher hurdle of course, but it ends up making the application development that much smoother because now you've got an engineering team that's building on both platforms at the same time and you're able to quickly experiment, get these things out to these ecosystems without having to implement them one-off in each platform.

**[00:21:24] JM:** That is a remarkable number of volume of code that you're able to share across the native mobile apps. Could we just zoom-in again on what are the mobile apps that we're talking about here? What are the native mobile apps that we're building in Shopify?

**[00:21:44] FT:** Sure. We have a bunch of different mobile services. Today, for React Native, we have three different apps that are under development. Two are in production and one is coming later this year. The two that are in production, one is the Arrive App. It's our consumer package tracking app that was originally built for iOS. It was built in 2018. We got excited about React Native at the end of 2018 and started a rebuild in 2019 to rewriting React Native.

After rebuilding it in React Native and then releasing it for Android, we saw a few interesting stats. One was of course the 95% code share. The other one was we saw a dramatic drop in crashes from deploying in React Native. So far, there are lots of reasons why that could be, but that's what we saw overall. That's the Arrive App.

The second happens in production is an app called Compass. It's an app that helps entrepreneurs learn about the entrepreneurial ecosystem and eventually build out a Shopify store. So that's also Android and iOS and that's also built in React Native. Then the third app that is not yet released for any platform is our all new point-of-sale.

As I mentioned, we have a point-of-sale software that allows merchants to have a retail experience. We announced in 2019 that we're going to be launching a brand-new point-of-sale in 2020. So both – We have an iOS version of that and a React Native version of that because we wanted to build them in parallel in order to understand how the platforms behaved in this environment. The React Native version of that app will also launch in 20/20.

**[00:23:27] JM:** One thing I'm trying to figure out is the difference between Shopify and Airbnb. We did this story about Airbnb and how they moved off of React Native because of certain difficulties that they had with the project. Shopify, to me, in some ways, it's very similar to Airbnb. You have people making purchases. This is mission-critical software that delivers high amounts of value to people. It's very sensitive in some sense. But one difference I can see is that, with Shopify, it's not – The apps you're building are not as much on the critical path.

I mean, in the case of the point-of-sale application, it's definitely still on the critical path, but it's a critical path that's not the – Well, I guess people buy things through the point-of-sale application, but maybe you could just give me your perspective on why didn't React Native work for Airbnb and why does it work for Shopify.

**[00:24:35] FT:** Yeah. No. It's a good question, because that is the most popular question we got when we announced that we're moving to React Native. I think a few things. Across all our mobile surfaces, they're definitely critical. You're right on the buyer side. That's mobile web. Probably, in terms of the millions of people who are interacting with it, probably the large majority will be on mobile web. Definitely for our merchants, use Shopify mobile. It's their lifeline to managing their store. Then retail, for sure, you need to have a point-of-sale that allows you to transact and look up products and get details and look at variance, etc. Definitely, there is mission-critical workflows happening through mobile.

I would say there're probably two or three big differences between us and Airbnb. I would say the first one is the timing. They moved off of React Native in 2018, and that's really when we saw the investments from Facebook really going in. A lot of the rebuttals to their article from Facebook when their very six-part article came out, was that everything that they were noticing or noting as problems in the platform were actually being built by Facebook.

Now, one response to that could be, well, you can always see announced things that are coming up, but when are they actually coming out and when are they going to be in the wild? That's one criticism, but definitely some of the things were already on the roadmap and being addressed. That's number one.

Number two is this cultural difference. This happens in any company when you've got people who really believe in one ecosystem over another. In this case at Airbnb, you have the native engineers versus the React Native engineers. This reminds me of, because at Shopify we use Ruby and Ruby on Rails, the Java people versus the Ruby people. It's very similar. If you're an engineer who really believes in a certain ecosystem of tool and that's the tooling you've become expert in and you don't feel like you want to try a new set of tooling or you don't believe in the new direction, then you might get stuck in your tool chain. I think that's in the conversations that had happened at Airbnb where the native engineers really didn't come across to React Native. If you look at – When I tweeted out our article, there was a lot of people replying to the tweet saying, "Hey, I was at Airbnb at the time, and a lot of it was cultural." That's number two.

Then number three is they did something, which I still think is quite difficult, although Facebook does it, which is they were building a Brownfield. A brownfield app is an application that has a combination of native, like native mobile and React Native. Whenever you've got two completely different tool chains in your mobile app, it's really hard to reconcile between the two for many reasons. One reason being, you really have to understand React Native well, and native development on mobile well, and then how they interact. That's one issue.

The other issue is going to be from the customer side, like a user. A user, it can come across jarring to the user when you're moving between native and React Native if the elements don't behave exactly the same or they behave differently between screens, especially transitions. All

of those combined together gave Airbnb a very poor experience in moving to React Native, and so they decided to move back.

**[00:28:03] JM:** Yeah, the separate tool chain's seems particularly brutal. I remember my conversations or my conversation that I had in that show. It just sounded like the build process was difficult. The release process was difficult. Networking was hard. Tell me about how that looks in a greenfield application.

**[00:28:25] FT:** Well, in a greenfield – Again, there's a few things different now, right? We are two years later. Two years later, and Facebook has reinvigorated the amount of work that they're doing on React Native. A lot has changed. There are many better tools. Even actually as recently as a year ago, just upgrading React Native was a real pain in the ass, right? A lot of people would say, "Oh my God! A new version of React Native will come out and actually the upgrade cycle was weeks of work," and that process has now largely gone the way of like it's a no-op where the upgrade can happen quite seamlessly.

Greenfield applications though, because of their nature of your building everything in JavaScript, you're trying to minimize the amount of native code, because again you want to have the effectiveness of having those engineers work in one tool chain. You want it to be cross-platform. So you were able to quickly build things in JavaScript now. Have them be deployed against both platforms and not worry about this interaction of, "Am I going to do now iOS or Android? Do I need to understand iOS or Android in a very deep level, at a very deep level in order to make this great application?" All of that goes away. How do I deal with transitions? Is it jarring?

Now, what's amazing about greenfield applications today as well as you can still build in native components, but that's quite different than having a native app and trying to expose React Native views in that native app. Today, if you're building React Native app and you realize that there's some sort of funky graphical UX that can only be done using native, you still have the leverage to drop into native and build that component natively, but that's quite, quite different than trying to take an existing native app and incorporate React Native into it.

That being said, one of the things that was interesting to me that got pointed out to me when I tweeted this article out, our article was about us moving to React Native, was the few engineers

who reached out to me to say, “Hey, we'd love to do a session with Shopify to show you how we think about moving brownfield apps to React Native.” Meaning, taking your native apps, incorporating React Native views into them and then eventually over time having them convert to be 100% React Native.

**[00:30:53] JM:** Did you meet with those developers who are cognizant of how to do brownfield application development?

**[00:31:00] FT:** Yeah. We’re still chatting with those folks. There isn't – Even when I approached them, there wasn't a tried and true. It was more of a conversation to figure out how we’re thinking about it. How they’re thinking about it. But I think this is a huge – This will be a huge boon for the mobile community. If we can come up with a playbook for folks to think about how they can potentially add React Native to their native apps in a way such that they don't run into the problems that Airbnb ran into.

**[00:31:31] JM:** Tell me about how development proceeds on these different mobile apps. Maybe you want to pick a case study and just tell me about like what did the developers on the team need to know. What does their development lifecycle look like? What is the release process look like?

**[00:31:54] FT:** Sure. I mean, I think the first thing to talk about though in terms of expertise is the teams who are building now are actually a combination of people from the web, so with React backgrounds, and mobile engineers, either Android or iOS. The teams have been – That we've put together at Shopify have been a combination, and I think it's important that you do get a team with that sort of combination, because there is something about understanding how mobile works. Understanding what experiences users expect that really makes a difference even when you’re building in React Native.

I don't want to discount anybody who's listening to this and is a mobile engineer and throwing their hands up in the air and saying, “Oh my God! I got to learn a new framework.” All of that experience is still hyper relevant in this world of React Native because of all the expectations of the user and knowing what's possible. There is a great blog posts from other companies who have moved to React Native and just said that it is really important to stay close to mobile.

I think the others thing that's interesting is the mobile engineers are learning a lot from the web engineers who are of the React engineers coming to React Native because they bring with them this reactive programming mindset that the mobile engineers may not necessarily be familiar with. You're learning how to build things in this new way. We've had multiple internal talks at Shopify for mobile engineers who say that they enjoy building React Native apps more than they ever did building native apps. I think that's important.

In terms of the development environment, like I mentioned earlier, it's we try to do as much as possible in the React Native world because that makes the entire lifecycle easier. I don't think the deployment process through the app stores is that different. One piece of advice I will give the teams thinking of adapting this is – And this is something that we didn't always follow, I think it's is a learning for us, is when you're building things in React Native, I think what happens to most teams is that they try to focus on one platform at a time. I think that in React Native world, that may end up being a mistake, because you may run into issues later on because you weren't thinking about the other platform. My advice to teams starting out is really start and deployed to both platforms from the beginning that'll not get you into a weird situation where you have to significantly make changes because you forgot about Android, for example.

**[00:34:30] JM:** Are there times where the React Native tooling does not play nice with the mobile platform and you end up with these strange bugs that are hard to interpret?

**[00:34:43] FT:** I still think that's probably one of the areas in which a lot more work has to be done in React Native. The debugging environment I would say is not as good as native. You definitely have a richer tool chain in native development. You're either in Xcode or Android Studio. So you've got a much more direct way to understand what's going on in the underlying system. In React Native, you've got a much deeper call stack to navigate, and so that can be harder.

I think that's probably one of the main concerns I get from mobile engineers when they're doing debugging. That can lead to some – In on some cases, some hairy problems. We saw, I think it was Discord who came out saying that they had to build new open source library to focus on fast lists. Lists are something that maybe forever will be something that people compare on

native and React Native because it is something that is quite optimized in these native platforms and not always as performant in React Native. So you have multiple implementations from different companies trying to build faster list implementations that recycle memory better. That's something that I don't think will maybe hopefully will go away at some point, but has not gone away yet in React Native.

But I mentioned, what I'm really excited about is the ability for these cross, like these teams of people who wouldn't necessarily work together. You typically would have a separate mobile team and a separate web team. Now, it's the same team.

[SPONSOR MESSAGE]

**[00:36:22] JM:** When I'm building a new product, G2i is the company that I call on to help me find a developer who can build the first version of my product. G2i is a hiring platform run by engineers that matches you with React, React Native, GraphQL and mobile engineers who you can trust. Whether you are a new company building your first product, like me, or an established company that wants additional engineering help, G2i has the talent that you need to accomplish your goals.

Go to softwareengineeringdaily.com/g2i to learn more about what G2i has to offer. We've also done several shows with the people who run G2i, Gabe Greenberg, and the rest of his team. These are engineers who know about the React ecosystem, about the mobile ecosystem, about GraphQL, React Native. They know their stuff and they run a great organization.

In my personal experience, G2i has linked me up with experienced engineers that can fit my budget, and the G2i staff are friendly and easy to work with. They know how product development works. They can help you find the perfect engineer for your stack, and you can go to softwareengineeringdaily.com/g2i to learn more about G2i.

Thank you to G2i for being a great supporter of Software Engineering Daily both as listeners and also as people who have contributed code that have helped me out in my projects. So if you want to get some additional help for your engineering projects, go to softwareengineeringdaily.com/g2i.

[INTERVIEW CONTINUED]

**[00:38:12] JM:** The Discord use case is particularly interesting to me. I should do a show about that. I don't know if you know anybody from the Discord –

**[00:38:20] FT:** I do. I know them. Yeah, I can introduce you.

**[00:38:22] JM:** Yeah, great. It's particularly interesting in light. I just did this show about Facebook Messenger and somebody from – There's a Facebook Messenger rewrite for iOS at least recently, and at its peak they were like 130 people working on this rewrite, which is just kind of amazing, but I guess that's Facebook for you. They did not use React Native at all in the rewrite of Facebook Messenger. I suppose the reason is because it's high-utility. It's utility, right? It's a communications tool. Maybe the networking constraints, the responsiveness, just the lightning-fast responsiveness you need. Perhaps the offline capabilities. These things make it so sensitive to use anything but native mobile.

On the other hand, Discord, is also a communications app. I realized you don't work at Facebook and you don't work at Discord, but do you have any perspective on why React Native maybe was not enough to satisfy the constraints of Facebook Messenger, or are there certain application domains that React Native perhaps is not ready for?

**[00:39:38] FT:** Yeah. It's really a question, and I did follow along, because obviously I'm quiet connected to Facebook and chatted with those folks about those. I think there's a few ways to think about it. I mean, one is you're right, it was a huge team that worked on Messenger. I think the other thing, I don't know if you got into this on the podcast. I haven't listened to it yet, is that Facebook is still really focused on Objective-C.

While mobile development, native mobile development on iOS has moved to Swift, and on Android has moved to Kotlin, Facebook remained on Objective-C, and I have it on good authority that some other popular iOS apps have also remained on Objective-C for performance reasons. So they really even feel that Swift, which is Apple's preferred environment for iOS development, they prefer to just still stay on Objective-C. They did very much care about

performance, but I don't think that's a reason, enough of a reason for them to have not looked at React Native.

Now, the comparison is I'm pretty sure Discord. They're not as big as Facebook, right? I think Discord has about 250 million users or something like that. I don't know how many of them are on mobile. But that that team is three. There're three engineers that built the Discord mobile app in React Native. I'm not sure. I mean, I would love for you to have that conversation with them. I'm not sure what all the differences are, but if you've ever played with the Discord iOS app, it is phenomenal. It is very performant. A lot of people talked about how they're so surprised that it's React Native.

Now, another example of one, and you could have this team on too, is the NFL. The NFL app is a top. I don't know the number. I'm going to guess. During the NFL season, top 10 app. They did a rewrite a few years ago in React Native as well for iOS and Android, and that has video and people who care about scores, because people who are betting. There is all sorts of performance constraints on that type of application and they also went React Native. Definitely, good to chat with those teams and to figure out how they made those decisions. But you're right. Like you said, maybe it's just Facebook and different companies have a different point-of-view. I mean, people would say that about Shopify. They're, like, "Why did we build that in Ruby?" They're like, "Oh! Because it's Shopify."

**[00:41:45] JM:** Do you have a generalizable perspective about what are the kinds of applications that React Native is not ready for?

**[00:41:57] FT:** In the blog post, I outlined a few areas. As I learn more about React Native, I'll start to probably update that point of view. One area where React Native is not good is when you've got hardware that has multiple cores, but each core is not that speedy. Because React Native is not fully multithreaded, I think it's got four threads are two. You don't actually get any advantages from having multiple cores.

If you're doing a lot of work with Android hardware, for example, and you are trying to lower the cost of your hardware or worry about battery life and you've got lower CPU speeds in exchange for more cores, that might not be a good tradeoff for you to do on React Native because you're

not to get the advantage of that power. You do have to then benchmark your application and worry about is that going to give you a great experience? That's something to think about.

The other one to think about is if you do have lots and lots of interactions with the hardware, because each connection to the hardware requires you to have a hardware module. If you're building specific hardware and you need to build a native module for each of those hardware components, you may run into issues where you're like, "Wow! I'm just really spending a lot of time in native code here. Am I getting the value out of React Native?" Again, I'm saying go with one or the other, but actually take that into your analysis.

**[00:43:29] JM:** What are the aspects of the React Native ecosystems that are changing the most right now?

**[00:43:36] FT:** Well, one thing I am very happy to see is that people are now coming out of the woodwork to talk about their experiences, right? I didn't even know NFL was React Native until I did my blog post. Then right after us, Coinbase went out and wrote their blog post saying, "We're also going to React Native." So you're starting to see these companies say, "Oh, by the way, we were doing this." I think Amazon tweeted back to us saying we are doing React Native. Of course, Microsoft. It was almost like a hidden club. People didn't want to talk about it. They've been getting these advantages, but because of the Airbnb – And it was a great detailed article. But because of that post, a lot of people felt like they couldn't go out and go public about their infrastructure. I think that's going away. I'm very excited. I'm excited about all the training and people learning React and react Native that feel like they can now go between mobile and web. I think that's great. You're seeing especially in these ecosystems were a lot of people are coming out of boot camps and they're learning JavaScript, it's the number one language whenever – As [inaudible 00:44:36] as well. If somebody asks you what language should I learn? I don't know if you would recommend JavaScript, but that's what I recommend now.

**[00:44:41] JM:** Of course.

**[00:44:42] FT:** Yeah. I think now you're able to have those same engineers be working on mobile apps. That's amazing and I'm excited about that. I'm excited about people contributing to open source. I follow all the GitHub repos and there is a lot of activity happening pretty often. I

think I'm excited about that. I'm excited about the community involvement. I'm really happy to see what Microsoft's been contributing back. I'm excited to continue our quest to contribute back whatsoever we learning in React Native. I love the company interactions.

Like I mentioned, I know the people at NFL. I know the people at Discord. I know the people at Amazon now, and at Microsoft, and at Facebook. We're becoming a real big community about this. It's not just pinning this on one company like Facebook.

**[00:45:23] JM:** A few specific questions about the development process. If I am developing a cross-platform React Native application and I'm just developing by myself, am I writing the React Native code first in iOS, or should I write it first in Android, or should I try to do both of them at the same time? Let's say I'm a React Native developer and I'm hacking together a cross-platform app by myself. Where should I begin?

**[00:45:57] FT:** Yeah. I think the way to think about it is you're going to write your application in React Native. So you're writing it in JavaScript and you're using React Native instead of React as your JavaScript libraries. But what I mentioned earlier is that you should deploy and test on both iOS and Android from the beginning, and that could even just be the iOS and Android simulators on your PC, and that will allow you to see how the app performs and behaves on those. Because I think what happens to some teams is they go down the path too far on one platform. It's probably okay if you end up not interacting with the hardware or using hardware modules or building any native modules. But if you do, you do any of those things, you may go down a path where you're like, "Oh crap! If I had just used this other platform earlier." Here's a good example. I heard this from another engineering team that I was chatting with I think it was over Twitter where they had locked themselves into a notification framework that was just iOS, because they didn't realize later on that that wasn't one that also work for Android.

I think it's just something that if you early on know you're going to be using it for cross-platform. Again, here's an interesting tidbit for you. Discord uses React Native for just iOS. It isn't the case that the only reason to use React Native is cross-platform. It is a really good reason, but in a Discord example, they do Android natively, but iOS using React Native.

**[00:47:23] JM:** Tell me more about how React Native has changed the structure of the mobile teams.

**[00:47:30] FT:** Sure. I mean, there's going to be a few things were going to see over the next few months, because as we build up these teams to be incorporated around React Native, we're starting to see a different level of experience. I mentioned earlier having JavaScript and React folks come to the team, that's been super helpful. Mobile engineers coming to the team. That's super helpful.

I think one thing we're going to notice a lot more is the deep infrastructure investments in React Native. One example is there are companies now that build like databases just for React Native. On mobile, you have like a SQLite, for example. You have realm, which is React Native database. I think you're going to see more investment into those areas and maybe even companies like ours or Microsoft or others might want to make a reinvestment in that area.

The teams will not just be focused on the user-facing side of mobile apps. They're also going to start thinking about the infrastructure side. At Shopify, for example, we have a mobile tooling team. We have a mobile foundations team that these two teams focus on the developer experience and also the core components that other teams can use. I think the structure of the teams is going to mimic what we have in the native development world. But because they're focus on React Native and the engineers can come from not only the mobile world, but also the web world, I think we're going to see a much more infrastructure investment there. I also think that the hardware modules that or already existing on native mobile will all become enabled in React Native over time. Like I mention, SQLite. There's already an effort to get SQLite working on React Native.

**[00:49:11] JM:** I'd like to get your perspective on just what it's like to be a VP of engineering in the midst of this dramatic change to how teams are working, the changes due to the virus COVID-19. Has it affected severely how you and your teams are working?

**[00:49:35] FT:** It's a complete 180 for us. We're a company that really believes – Shopify is a company that really believes in that like great collaborative work requires some level of in-person work. For us, we definitely have teams that have always worked remote definitely in

production engineering, but we also have teams that had to rediscover a new way of working due to COVID-19. We've been remote now for – Actually I think it's two weeks tomorrow. We've been remote. We're learning new cadences. We're learning new ways to use Slack. We're learning about how to use Hangouts and Zoom in a more effective way to be social. Not just talk about work. We're figuring out ways just hang out and have fun. We're doing lunch is over Zoom. I mean, I have almost – Every night, I've got like a different dinner over Zoom with people that we were going to meet with. I think, yeah – Actually, you might find this interesting. I have a dinner tomorrow night with like 18 other VP engineering's that are all over the world and we're all having dinner at 9 PM Eastern over Zoom to have a conversation just like we would in-person. I think everything has changed.

Now, one thing I like about this is that I'm a big fan of like anti-fragility. I'm a big fan of trying new ways to do something, because I think there will be learnings that we will take back even though we will eventually go back into our offices that I think will make the team stronger. We were already a distributed team, but we were not a remote team.

**[00:51:05] JM:** Can you just tell me more about what is the function of a VP of engineering at Shopify? I think VP of engineering is a role that can encompass a lot of different things. I just like to know what you're focused on right now and what the mission of a VP of engineering is more generally.

**[00:51:26] FT:** Sure. It's a good question. I think it is actually quite different in every company. For me, at Shopify, focus on a few things. One is air traffic control. There're lots going on in the company, and my role is to unblock people if they do get blocked, and that could be resource constraints. that could be like helping people, like if you're resource constrained in one area, maybe I can get resources from another. A lot of my time is spent on recruiting. I spent a lot of time with people internal and external at Shopify thinking about what they could be doing differently such that they can have a higher effectiveness at Shopify. If that's external, that means maybe I can convince them to come work with us. If it's internal, maybe I can move people around for things that they're aligned to working on.

There's a lot of time spent on understanding the craft. My background is from like Pivotal, Pivotal Labs. A few companies ago, my company was acquired by them. I am a big fan of

understanding when process can enable things and when process can hurt things. I'm a big fan of trying to figure out where the roadblocks are and helping understand by asking questions. Is there a different way to do something?

I don't know. I mean, any day could be different. Like today, for example, most of my day is filled with meetings. Doing lots of one-on-ones with folks. Thinking about our product roadmaps. Doing recruiting meetings, which means like meeting people external and trying to be always available on Slack to see if I can unblock anybody on any issues that they might be having.

**[00:52:56] JM:** Well, just to close off since we've been exploring React Native. Tell me what your predictions are for how it's going to affect Shopify going forward.

**[00:53:06] FT:** I think I'm very excited to see how much more effective we can be as a company developing a mobile. What that means for me is really focusing on the speed at which we can build things. The speed at which we can experiment. How different parts of the organization can contribute now that we've gotten more of a shared background experience. Like I mentioned, we're a company that develops lots of software for the web. Lots of people who know React. Having those people come across and contribute to mobile.

I'm also very excited on the mobile tooling and foundation side to see how much more effective we can make on mobile ecosystem. Then lastly, I can't wait to contribute back. I really want to focus Shopify on owning some of the React Native community modules. Spending a lot more time with Facebook, Microsoft and others on that ecosystem and making sure that we can make this a much more effective environment for mobile engineers all over the world and building anything that they want to build for their users.

**[00:54:12] JM:** Farhan, thanks for coming on the show. It's been really great talking to you.

**[00:54:14] FT:** No problem. Then just to sign-off, we are always hiring at Shopify, and actually hiring a principal engineer in mobile.

**[00:54:21] JM:** All right!

**[00:54:21] FT:** Yeah. Anybody who's interested can check out shopify.com/careers.

**[00:54:26] JM:** All right, great! Well, thanks, Farhan, and I'll talk to you soon.

**[00:54:29] FT:** Thanks, Jeff.

[END OF INTERVIEW]

**[00:54:39] JM:** Over the last few months, I've started hearing about Retool. Every business needs internal tools, but if we're being honest, I don't know of many engineers who really enjoy building internal tools. It can be hard to get engineering resources to build back-office applications and it's definitely hard to get engineers excited about maintaining those back-office applications. Companies like a Doordash, and Brex, and Amazon use Retool to build custom internal tools faster.

The idea is that internal tools mostly look the same. They're made out of tables, and dropdowns, and buttons, and text inputs. Retool gives you a drag-and-drop interface so engineers can build these internal UIs in hours, not days, and they can spend more time building features that customers will see. Retool connects to any database and API. For example, if you are pulling data from Postgres, you just write a SQL query. You drag a table on to the canvas.

If you want to try out Retool, you can go to retool.com/sedaily. That's R-E-T-O-O-L.com/sedaily, and you can even host Retool on-premise if you want to keep it ultra-secure. I've heard a lot of good things about Retool from engineers who I respect. So check it out at retool.com/sedaily.

[END]