EPISODE 366

[INTRODUCTION]

[0:00:00.4] JM: React Native allows developers to reuse components from one user interface on multiple platforms. React Native was introduced by Facebook to reduce the pain of teams who were rewriting their user interfaces for web, iOS, and android. Nader Dabit hosts React Native Radio, a podcast about React Native. Nader also trains companies to use React Native through his company React Native Training.

In this episode we explore what a developer can and cannot do with React Native and the state of the ecosystem, when a developer needs to use native APIs instead of React Native, and some speculation on the future of React Native. This episode is a good practice for tomorrow's episode about React Native interfaces with Leland Richardson of Aribnb. In that episode we're going to dive deeper into how React Native works and just how big of a change it could be for cross-platform developers.

If you are looking for some specific episodes about different topics in software engineering, check out our new topic feeds. We have these new feeds in iTunes or wherever you find your podcasts. We've sorted all 500 of our old episodes into categories like business and blockchain, JavaScript and machine learning. We have a greatest hits category that I recommend. It's the best episodes of software engineering. If you haven't been with the show since the beginning you may find these feeds useful for figuring out which old episodes to listen to because there's a lot of evergreen content in Software Engineering Daily. Whatever specific area of software you're curious about, we have a feed for you and you check the show notes for more details.

[SPONSOR MESSAGE]

[0:01:57.4] JM: Spring is a season of growth and change. Have you been thinking you'd be happier at a new job? If you're dreaming about a new job and have been waiting for the right time to make a move, go to hire.com/sedaily today.

Hired makes finding work enjoyable. Hired uses an algorithmic job-matching tool in combination with a talent advocate who will walk you through the process of finding a better job. Maybe you want more flexible hours, or more money, or remote work. Maybe you work at Zillow, or Squarespace, or Postmates, or some of the other top technology companies that are desperately looking for engineers on Hired. You and your skills are in high demand. You listen to a software engineering podcast in your spare time, so you're clearly passionate about technology.

Check out hired.com/sedaily to get a special offer for Software Engineering Daily listeners. A $600 signing bonus from Hired when you find that great job that gives you the respect and the salary that you deserve as a talented engineer. I love Hired because it puts you in charge.

Go to hired.com/sedaily, and thanks to Hired for being a continued long-running sponsor of Software Engineering Daily.

[INTERVIEW]

[0:03:27.6] JM: Nader Dabit is the host of React Native Radio. Nader, welcome to Software Engineering Daily.

[0:03:33.8] ND: Thank you for having me.

[0:03:34.8] JM: You're the host of React Native Radio, which is a podcast about React Native, and you run a company called React Native Training. You have your ear to the ground on how people are using React Native. The most recent show that we did about React Native was about Expo. We talked about the future of React Native, and we can touch on that a bit today. From you, I'd really like to understand the present of React Native. What are the types of projects that developers are building with React Native today?

[0:04:12.9] ND: There is really a huge range of things that people are building. First of all, our company, React Native Training, we do a lot of on-site consulting and, of course, training for large software companies, Fortune 500 companies. We also work with startups, and it's just been very existing to see the type of companies that are picking React Native up, companies

that you would think of is not really needing to worry about what a lot of people use React Native for, and that is for maybe cost savings or maybe they want to develop across an array of platforms, and they're a startup, so they're kind of limited on developers so then React Native is seen maybe a cost cutting measure in some areas or by some people I guess I would say.

We're actually seeing companies that have no worries about money, no worries about anything. They're taking React Native on and they're using it and they're actually using it for many of their newer projects. if they have a lot of brownfield projects, they may stick to native, but a lot of their newer projects, they're actually implementing React Native.

In general, I would say, anything that's not really related to gaming. Pretty much anything across the board, but I would say one thing I've seen people kind of not use React Native for is anything that needs a lot of animations as far as what would you think of, I guess, with games and stuff like that. As far as any type of business applications, social applications, pretty much anything people are using React Native for.

[0:05:41.9] JM: When you talk about these games, these high-performance heavy animation, heavy visual applications, what are the limitations of React Native that prevent people from writing a game, a really rich experience?

[0:06:01.0] ND: There just isn't a ton of platforms that are kind of built on top of React Native for gaming. For example, if you want to build a game, Unity is such a robust and mature platform. From Unity, you can basically export to iOS and android, so it's kind of like there's just a lot more tooling there and it would just be quite a bit of work. Now, there have been a few games that have been built with React Native, but nothing kind of too crazy, I guess I would say.

[0:06:32.1] JM: What are some examples — Let's say I'm just writing an application like a Netflix or a Facebook on React Native, where are the places where I have to write native code, where I'm going to have to duck down beneath the React Native layer into the actual native code. What are the different areas of the iOS stack or the android stack that are not yet accessible via React Native?

[0:07:03.0] ND: I would say as of now, most of everything that you need for pretty much 95% of use cases is already there in React Native. It's part of the framework. I would say about a year ago, we were going into the native code and kind of exposing APIs and things like that much more than we are today.

Really, it would be more like a very specific use case in your company, say, that you need a certain API that isn't exposed that is kind of a one-off type of thing, I would say. Most of the things that you're going to need are actually already available. I see more and more as we go on, the need to actually dig into a native side is just kind of going away.

[0:07:42.2] JM: If I'm just writing an iOS app these days or if I'm just writing an android app these days, how does the experience of React Native compared to the native experience? Should I just be writing all of my iOS apps in React Native?

[0:07:59.1] ND: This is a question that comes up quite a bit, especially we do this on-site training and on-site consulting. We're working mainly these days with native developers, because a lot of the people that are coming in to React Native that are seeing the most benefit out of it, or companies that have iOS teams and android teams and these teams are working great on their own, but when they are trying to create maybe a cross-platform experience that kind of mimics the entire UI both on iOS and android, they're having issues.

I would say if you're building an app that only needs to run on iOS or an app that only needs to run on android, I would stick with the platform, the native platform. If you do need to build cross-platform, use React Native. The benefit, I guess, mainly that we're seeing and what mainly the native developers are really liking about React Native, for the most part, it's going to be kind of like the hot module replacement and the live reloading and the developer experience as far as being able to build and iterate and see results quickly without having to wait for that long compile time.

A lot of these developers are used to having to wait anywhere between 15 seconds to maybe even a couple of minutes for their app to compile, but with React Native it's instant and they're able to work faster, iterate faster, build components that are kind of — Instead of working in their own language as far as what they're used to doing, as an iOS developer, maybe working with

Objective-C or Swift, and they enjoy Java or whatever. Now, they're kind of being placed on teams where they're working on a component or a feature and they implement that feature and that's kind of their thing as supposed to implementing a feature and a language, they're implementing a feature across the entire app.

[0:09:44.0] JM: What's the relationship between Apple and the React Native community?

[0:09:49.6] ND: I wouldn't say there's that much of a relationship. They've made it pretty clear about a few things that are React Native specific, like the idea of updating your app, baptizing the app store and they've kind of made it clear that that's okay as long as we're only using JavaScript. The relationship between Apple and React Native — I would say Apple is one of the few companies that I've seen that have not kind of publicly adopt their React Native, use it within their company, at least that I know of. I know they're using a little bit of React, but I'm not sure if they've actually done any React Native stuff.

[0:10:22.0] JM: Do you get a sense that there is a disdain from Apple or you just get this, like they just ignore it. They don't care. They're living in their own Apple world.

[0:10:32.9] ND: Looking at it from a business perspective, it seems like they would want to have more high-quality apps come into the app store so they can make more money. If React Native kind of allows that, it seems like it would be good for them in the long run.

[0:10:48.7] JM: What about Google? Google's got android. What's their perspective on React Native?

[0:10:53.9] ND: Google, they have, of course, Angular. Angular has something similar to React Native called Native Script. They seem to kind of be focusing on that Angular side. I haven't really seen a lot of movement as far as React Native with Google at least that I know of. I know they've developed this new framework they're kind of talking about, and it should be kind of getting a little more [inaudible 0:11:13.9], called Flutter, I believe it's called. Flutter kind of allows you to do something similar to React Native, but it's using Dart. That's going to be interesting to watch. I don't know enough about to really speak of it, but it's something to kind of take a look into it if you're, I guess, interested in this sort of stuff.

[0:11:29.7] JM: It's so interesting to watch the difference in how certain open-source technologies are perceived, you look at Kubernetes for example, and part of the open-source community probably would love for Amazon to adopt Kubernetes wholeheartedly, but because there is a certain threat or at least a perceived threat to their business, the threat to AWS by Google Cloud having really good knowledge of how to run Kubernetes, there is not at least, as far as I can tell, much public adoption of Kubernetes from AWSs side. It seems to be the same thing with React Native, you have this highly popular way of writing mobile apps and maybe the less popular ways of doing cross-platform mobile apps or Google coming up with just brand new way to break cross-platform mobile apps. I don't know. In a perfect world, for the consumer, it would seem like there would be more adoption for React Native, but of course we don't live in a perfect world.

[0:12:34.0] ND: Yeah, Dart has been something that they've kind of been trying to push for a while now. if you remember a few years ago, they made a few big launches and they were kind of pushing Dart and it just hasn't seem to be picked up. I'm kind of prone to kind of talk about React Native in the sense that the reason that it's become so widely adopted and it's become so successful is that it just uses JavaScript. It has a great engineering feat and it's amazing to work with and it's great. I believe that it's gotten to be that because there's been so many developers that have been able to kind of use it, contribute back to it, give feedback about it, because they're just so many JavaScript developers. The problem is they're just not a lot of Dart developers. There's not a lot of developers in some of these other languages that other things are coming out in like, say, C# for Xamarin. Xamarin is great, but there're just not a number of C# developers as there are JavaScript developers.

I would kind of contribute part of React Native's success to the fact that a lot of these traditionally web developers are coming into it, contributing to it, being able to pick it up and write apps with it. Then you also have the native developers that are kind of using it. It's kind of like that critical mass, and I would kind of attribute much of the success of React Native to that.

[0:13:44.2] JM: When Apple comes out with something like ARKit, which is this framework for augmented reality recently, how does that kind of announcement affect the React Native community, or does it affect the React Native at all?

[0:14:01.0] ND: I'm kind of curious to learn more about ARKit. Do you have any information about it, like what language it's built in and all of that?

[0:14:07.9] JM: No. All I've seen is the Splash page.

[0:14:10.3] ND: I would say we're always kind of — Most developers are always kind of like tuned in to what's going on in the world of software development. I think that kind of lends itself to a few different things. One is what is the next platform?

Right now, the big platform that is kind of dominating is mobile, but that's not always going to be the case probably. There's going to be something else that kind of comes out. For me, at least, I'm not sure I can speak to the rest of the community, but I'm always kind of interested in what's going to be kind of the next platform. I think we're years away from kind of really knowing and seeing that there's AR, there's VR, there's a handful of other things. You're talking about maybe the idea of the apps that are Echo and Amazon Alexa where you're kind of having these voice-based applications where you're interacting with voice.

I think we've yet to see the next platform that has been as successful as mobile, and I think that kind of lends itself, I would say, to that discussion in my opinion. Of course, you can build theoretically AR and VR with React Native. I believe with React VR, you can actually already build some stuff like that with React Native. I've yet to look into it.

[SPONSOR MESSAGE]

[0:15:27.4] JM: Angular, React, View, Knockout, the forecast calls for a flurry of frameworks making it hard to decide which to use, or maybe you already have a preferred JavaScript framework, but you want to try out a new one. Wijmo and GrapeCity bring you the How to Choose the Best JavaScript Framework For Your Team e-book.

In this free e-book, you'll learn about JavaScript frameworks. You'll learn about software design patterns and you'll learn about the advantage of using frameworks with UI libraries, like Wijmo. You'll also learn about the basic history and the purposes of JavaScript's top frameworks. You'll

also learn how to integrate a WIjmo UI control in pure JavaScript and in some of the top JavaScript frameworks that we've already were talked about, like Angular, React, View and Knockout.

Wijmo's spec method allows you to determine which framework is best suited to your project based on your priorities. Whatever those priorities and your selections are, you can learn how to migrate to a new framework in this e-book. Best of all, this e-book is free. You can download your copy today to help choose a framework for your work at softwareengineeringdaily.com/grapecity.

Thanks to GrapeCity for being a new sponsor Software Engineering Daily, and you can check out that e-book at softwareengineeringdaily.com/grapecity.

[INTERVIEW CONTINUED]

[0:17:10.4] JM: Whens something like ARKit comes out, and I guess you and I are both sort of speaking offhand it because we don't really know what it does or at what level of abstraction it works at. I don't know. Maybe you're able to speculate. Is this the kind of thing that React Native developers would build on top of? Would you build a React API on top of it to be able to work with the ARKit, or is this the kind of thing where you would sidestep React and write some native code that would interface with the ARKit because this is like an iOS-specific thing. It might not necessarily be on android. There might not be like a similar thing on android. Are there best practices for what the open-source React Native community would do to be able to access this AR functionality in React Native?

[0:18:05.9] ND: So far what we've seen is whenever there's a new platform that kind of comes out, someone kind of builds an abstraction to let us build on top of that new platform using React or React Native. For example, when the Touch Bar came out, within a couple of days or maybe a couple of weeks, there is a React Native Touch Bar, so you're able to basically use the information and the knowledge that we already have to build React Native and start building Touch Bar outs.

I believe for the most part we're going to kind of see that continue if it's possible. If someone can kind of build a React Native ARKit abstraction on top of ARKit and make it actually work in a decent way, I guess you would say, then that's kind of probably going to happen. Really, the core idea at least when they announced React Native was this idea of learn once and build anywhere as supposed to what we we're having, I guess before React Native, with some of these other hybrid technologies was build once, run anywhere.

The idea of learn once and build anywhere basically lends itself to say, "Let's learn the core concepts behind React and React Native and then we kind of build across all these platform without having to relearn a bunch of stuff. If it continues to kind of be the way we've been seeing it happen, then that's probably what's going to happen if it's possible.

[0:19:22.6] JM: If I build something on top of ARKit, for example, or just some other software package, is there a performance penalty or is there no performance penalty because it gets translated it from a React component into the native code at compile time?

[0:19:43.6] ND: I think you'll never be able to match native performance with any abstraction, because just the idea of an abstraction, we're kind of building on top of what we could just kind of be building in. Yeah, there's definitely going to be a little performance. That can be mitigated to almost nothing, I would say, if you kind of are really aware of what you're doing and you kind of know the best practices. You'll never, in my opinion, get to the point where you're having the exact same native performance.

It will be so minimal that maybe that the user will never know the difference, but if you're kind of benchmarking things, then you will see, in my experience at least, a little performance. You'll talk to people that kind of will say that that's not the case and you'll see people that kind of maybe take that into an extreme and say it will never be even close to native. In reality, what I've seen so far has been people are building really high-performant apps using React Native that are kind of undistinguishable, really, from native apps. If you kind of benchmark these things using some of the benchmarking tools that we have and you'll see that there is a slight performance hit.

[0:20:53.3] JM: Right. If there are — There's a certain app I use that I'm pretty sure is written in React Native. I don't want to name the name of it, but when it does some very simple — Like loading a simple list view, for example, there are some certain stutteriness that seems like if it were a purely native app, it would not experience, but I don't know if that's the React Native or if that's just a poor implementation.

[0:21:24.6] ND: A lot of the times, stuff like that is a poor implementation for something like that.

[0:21:27.1] JM: Okay.

[0:21:28.1] ND: Yeah.

[0:21:29.7] JM: Okay. Then I should speak to that. Let's say ARKit, it comes out and somebody does build the React AR layer on top of ARKit and then android comes out and android also has their android AR thing and it's slightly different. It has slightly different native APIs and then the React Native community says, "Well, we want to build a set of APIs in React Native for interfacing with augmented reality," and maybe there's a disconnect between how Apple views AR and how android views AR. How would the React Native community compromise in that case? I realized this is very much a prognostication, but I'm just trying to get a handle of how you build a consistent layer on top of different inconsistent APIs.

[0:22:24.4] ND: That's going to be super interesting to kind of watch. If you kind of really think about it, the inconsistencies between iOS and android are pretty startling. If you actually learn iOS development then you're going to learn android development. The way they've kind of combine these APIs into a single consistent React Native API is pretty amazing.

I feel like there will never be a consistent, a 100% consistent API because there are going to be things. Even with React Native, you have things like tap Tab Bar iOS that are not on android, and you have things that are just never going to be there and you kind of have to code your way around those. When you hear about code reuse being somewhere in the range of 75% to 95% for React Native cross-platform, that's where that 5% to 25% comes in where we're kind of seeing those disagreements, I guess you would say, and implementations.

I've been pretty amazed so far from what we've seen from React Native as far as being able to build across all these platforms using that consistent single API. I feel like as long as it's kind of within the realm of being possible, then someone will kind of build it and make it available.

[0:23:41.8] JM: You run this training business, it's called React Native Training, right?

[0:23:46.7] ND: Yes.

[0:23:47.2] JM: Okay. I did a show a while ago about companies that use Ionic, and often times these companies that use Ionic, Ionic is a cross-platform way to build — You write an Angular app and it gives you a cross-platform app that works on Android and iOS. The people I talk to about Ionic said that there's a lot of people who — A lot of large companies that will adopt this, because let's say they are some kind of warehousing company, or like John Deere Tractor. I don't know if John Deere Tractor was one of the actual examples, but I think of John Deere as this giant enterprise.

It's not exactly a software company, but they've got all these technical problems, like you've got a bunch of engineer — Or you've got a bunch of warehouse workers and they're trying to synchronize different things and they're building tractors and what not. Everybody's got a mobile phone and they've got some domain-specific apps on their mobile phone that help them manage the inventory in the warehouse, maybe it helps them notice things around the factory. At a giant company like John Deere, you might have a bunch of different domain-specific mobile apps. They're just internal mobile apps and you don't need to really worry about UX, for example. These are the kinds of companies that love Ionic, because you're just building these simple apps with simple list views that are just kind of doing crud stuff with a database somewhere and the UI really doesn't matter.

When you're training these companies to learn to build React Native apps, how many of them fall into that same category where it's like some big company and they just want to build cross-platforms apps, because they have all these employees because — And employees have iOS phones, they have android phones. If you want to build internal apps for your entire employee-base, you don't really want to worry about building separate android and iOS apps. Are a large

percentage of the companies that you're teaching React Native, are they these kinds of large enterprises?

[0:25:55.5] ND: It's hard to kind of put everyone into a single kind of niche or whatever because, really, I've worked with literally the top — Two out of the top five technology companies in the world all the way down to startups and everything really in between. For example, I just left Palo Alto last week, spent three days with this company. A few guys that had left Google have created this startup. They have good funding. They have an amazing back-end team with artificial intelligence engineers and they've created this amazing APIs. They have six apps that are ready to go in the back-end, but they are bottlenecked on the front-end because they only have a handful of native developers and they've spent a few months and they've only kind of built maybe a single app and a single platform.

We come in there, we spent three days. We basically transformed many of their iOS and android developers into React Native developers and they also had a couple of wed developers that kind of set in on the training as well. Now they're all able to contribute to build React Native and cross-platform apps. Now, with the cost of engineers being so expensive especially out there in Silicon Valley and really in any big city, it's not only expensive but it's hard to find good ones. When you do find a good developer, you want to kind of maximize their efficiency.

Basically, what React Native is allowing these companies to do is take their existing knowledge and kind of sharpen it and accelerate it and kind of start building with React Native using a lot of their existing knowledge along with this new React Native and they're building the exact same applications they are building before. The UI is amazing. We're not just building list views, we're building really nice user experiences with animations and everything that you would kind of want from an enterprise app or a high quality app. One of those apps where you kind of are playing with it and you're using it and it just feels really good. That's kind of the target for React Native. I think that's kind of why React Native is done a little better than Cordova, and I feel like Ionic is amazing and it's a really great framework and I used it quite a bit before I kind of got in React Native, and I feel like React Native is maybe one step or a few steps ahead of Ionic just because with Ionic, you are kind of limited to web technologies. You have to worry about browser issues and things like that. You're working with a DOM. With React Native, you're actually working with real native components.

The enterprise companies I've worked with, these companies — I'm just kind of blown away when I hear the projects that they're working on. They're literally taking billion dollar applications that are already generating revenue and they're transforming these into React Native. They're taking new projects and they're building these using React Native. They're not building these using iOS or android developers or using native iOS and android platforms. They're building these.

A company that I've done quite a bit of consulting with, that is one of these top five tech companies, is building all of their new native apps using React Native. They're not affiliated with Facebook in anyway. It's kind of been very interesting. It's very promising at least from my company, because we are in that space and we're kind of doing that type of training and it's very interesting to kind of see this kind of uptick in interest and adoption.

[0:29:23.8] JM: When you're training these companies to use React Native, you tend to start with the fundamentals. What are the fundamentals of React Native?

[0:29:35.5] ND: The fundamentals of React Native are really the fundamentals of JavaScript. We kind of start of by walking through a generated React Native project just kind of going through the folder structure and kind of looking at the code. Then we literally jump into a couple of hours of JavaScript. We're doing a combination of JavaScript basics but we're also doing ES6 and ES2015 fundamentals. We're kind of reviewing not only basic JavaScript stuff, but we're using these syntax that you're going to be seeing in all of the documentation, all of the open-source projects, all of these stack overflow answers. We're kind of explaining what this new ES6 syntax is.

That's kind of the fundamentals of React Native or, really, a combination of JavaScript and React. So because React Native is built on top of React, you still have these existing things that are React-specific, like lifecycle methods, like component creation, like data management, and props, and state. We kind of do an entire day almost of going over all of the fundamentals of React and JavaScript. Once you kind of understand all of that, you're able to get up and running pretty quickly.

I think the reason that we've had such demand, really, is because a lot of these developers are used to coming from ecosystems like Java and iOS that are a little more consistent and they're a little more cohesive. Whereas JavaScript and the web technologies in general are kind of like the Wild Wild West. You go online to find some answers, some documentation. You'll find 15 different answers, and it's kind of hard to kind of pinpoint what is the best practice without wasting a lot of time. We kind of distill all of these best practices, all of these things into a single two or three day boot camp type of training and kind of get everyone laser-focused on what exactly you need to know and kind of — The most up-to-date practice isn't the most up-to-date libraries and all of that stuff.

[0:31:34.0] JM: In React, you are building apps out of components. Give an overview of the React component lifecycle, because I know this is something that you tackle in your training operations.

[0:31:47.6] ND: Yes. There is kind of two different ways to create a React component. One of them is a statefull component, and you normally do that using a class, a JavaScript class, and then there's a stateless component. A stateless component is basically creating a React Component or a piece of UI and functionality just using a JavaScript function.

As far as the lifecycle methods are concerned, you basically are using a React class, using a JavaScript class, really, and you kind of hook into these components. When you create a new component, you kind of have all of these lifecycle methods that are triggered and you kind of hook into them if you would like. There are things like component will mount, component didn't mount. You have render, and you're kind of able to hook into APIs and you're able to do different things within that lifecycle.

We kind of will dive into exactly how all these work and kind of when you should use each of these and when you should not use each of these, I guess you would say.

[0:32:42.8] JM: I did a show with Tadeu Zagallo who works on the React Native team, and he described this bridge between native code and JavaScript. This is essentially what facilitates the JavaScript that you write for React Native, how that gets translated into a native code or how it communicates with native code. Describe how that bridge works.

[0:33:06.5] ND: Yes. Basically, we're able to kind of go into — Say, if we want to build something and expose an API from iOS, we'll create a header and an m-file and we'll write our functionality and we'll basically export that using a method called RCT, I believe, export. Then you're able to just to require whatever functionality or whatever UI you need from within your JavaScript class or your JavaScript file. You will call RCT export and you'll pass in whatever that ever that you've created and then you're basically able to just import it from your JavaScript just like you would a normal JavaScript file. It's pretty interesting that it's pretty simple to use once you kind of understand it. I believe that's kind of really what I've seen so far, at least, the bottleneck for web developers and JavaScript developers whereas supposed to an iOS or an android developer, they kind of shine when it's time to kind of start digging into the more complex stuff, which is bridging.

For us web developers, because that's what I am, I'm traditionally a web developer that kind of I'm getting into React Native as supposed to a native developer getting into React Native, understanding that native side is very very valuable. You're able to kind of quickly implement things as a native developer that a web developer or a JavaScript developer probably would take years to learn, because you already kind of understand the underlying building blocks and things like that.

In general, the best practices and based on the people that kind of create and maintain React Native, the whole idea really is to kind of stick to JavaScript as much as possible and kind of not go into the native side unless you absolutely have to. That's just kind of something to keep in mind if you're listening and you're kind of wanting to learn more about it. It is very simple to kind of hook into the native side.

[0:34:49.9] JM: You write that lists are very important in React Native, and list views are fundamental to apps, but they seem like such a simple subject to me. What is so complicated about lists that you have to spend a lot of time on the subject of lists in your training sessions?

[0:35:11.8] ND: The basic idea of a list is very simple. You just have an array of items and you kind of just spit those out into your UI. When you're actually dealing in a real business case scenario, there's a lot of other things that are kind of involved. First of all, say, you have an

infinite list, like how many items do you want to show per page and what performance benefits you're going to get if you kind of understand how many items it should show versus what performance hit you're going to take and you kind of just spit out a million items, and how do you handle that?

Also, there's things like you're going to have interactions with these lists, maybe you want to have an on-press method or maybe you want to have different things that are rendered based on whatever is kind of in that dataset. I wouldn't say it's kind of a hard thing to understand. It's one of those things where there's an 80-20 rule. If you kind of fundamentally really brought and kind of can build and use list very quickly and efficiently, if we kind of spend a lot of time going overall the different ways there are to do that and how to handle all these issues, you can kind of get up and running quickly in a real-world scenario and be very productive.

[SPONSOR MESSAGE]

[0:36:30.4] JM: At Software Engineering Daily, we need to keep our metrics reliable. If a botnet started listening to all of our episodes and we had nothing to stop it, our statistics would be corrupted. We would have no way to know whether a listen came from a bot, or from a real user. That's why we use Encapsula to stop attackers and improve performance.

When a listener makes a request to play an episode of Software Engineering Daily, Encapsula checks that request before it reaches our servers and filters bot traffic preventing it from ever reaching us. Botnets and DDoS are not just a threat to podcasts. They can impact your application too. Encapsula can protect your API servers and your microservices from responding to unwanted requests.

To try Encapsula for yourself, got to encapsula.com/sedaily and get a month of Encapsula for free. Encapsula's API gives you control over the security and performance of your application. Whether you have a complex microservices architecture, or a WordPress site, like Software Engineering Daily.

Encapsula has a global network of over 30 data centers that optimize routing and cacher content. The same network of data centers that is filtering your content for attackers is operating as a CDN and speeding up your application.

To try Encapsula today, go to encapsula.com/sedaily and check it out. Thanks again Encapsula.

[INTERVIEW CONTINUED]

[0:38:15.8] JM: Yeah. The difference between — React Native and Kubernetes are the two open-source projects I've probably spend the most time discussing these days because they have so much momentum behind them, and React Native is basically on the front-end and Kubernetes is on the back-end. Kubernetes is a — I see a little more of a direct route to monetization for companies that are working on open-source it because it's like a platform and you can build higher level stuff on it, you can build higher level offerings that you can basically charge for an arbitrage cloud storage or cloud networking, like different things where you can charge people based on the amount of resources that they consume, and that's a really good business because you get margin off of those resources that you're consuming.

With React Native, it's a little bit different because you're not really — Whatever you're building, you're not really turning it into a SAS product, unless you're like maybe Expo. Expo is kind of — You could see that morphing into a SAS product.

That kind of makes sense to me that the angle that these companies are taking where, "How are we going to monetize our expertise in React Native without building React Native apps our self?" We do training. We do educational resources. We do outreach. We write open-source software as a way of getting leads and then talk to people.

That makes a lot of sense to me. It's too bad that I think part of the reason why Kubernetes is moving much faster — I don't know if it's moving faster than React Native, but I think the amount of attention that Kubernetes has is significantly higher than React Native because there is some bigger monetization opportunity there. If there were some bigger monetization opportunity, then perhaps there would be more of a direct route to making money off of it.

You see these sort of consulting style routes or training style routes basically because of the monetization hurdles for people who are experts. Does that resonate with you what I just said?

[0:40:23.5] ND: Yeah, totally. The scalability of something like Kubernetes or whatever — I'm not sure if I said that right, is on a different ballpark, really, than something like React Native just because you're looking at the basic business fundamentals of something like that, how that could happen.

When you're talking about monetization, it really kind of depends on what the focus of the company is. If you want to be a billion dollar company, it may be tough to kind of build something just in the — I would say, in the sense of what we're kind of doing with React Native training as far as an open-source project and make that a billion dollar product.

If you are just kind of looking to kind of make a living and do what you love to do and kind of help the community, or even build an agency which could possibly make a lot of money. I believe this whole idea of building like an open-source repository and building a lot of open-source stuff and giving it away and using that as a way to market yourself, market your expertise, market you as a leader in that space, similar to how Infinite Red and Formidable Labs are both kind of doing in the React and JavaScript space, React Native space.

Formidable Labs has just an array of really great open-source projects that they've kind of created and put out there. They have a few different ones. I believe one of them you can make slides with — I'm completely forgetting the names of all the stuff that they've done. I think it's Spectacle, where you can create slides using JavaScript. It's really nice.

Anyways, you look up a few different consulting companies for example, you want to build an app. You run across Formidable and then you run across a few other one. You see Formidable has like all of these repos and they've done all these awesome stuff, and you can kind of see their code and they are obviously leaders in a certain space. You're probably going to go with them. It makes a lot of sense.

Then you have companies like React Training with Michael Jackson and Ron Florence, and us, React Native Training and a few other companies that are kind of doing a similar thing in the open-source space with training and consulting.

For us, I feel like it's a good way for us to kind of instead of getting paid directly to build things like React Native elements, which we have a UI library for React Native, or create XPF, or any of the other repos that we have that kind of benefit React Native developers. Instead of getting paid directly to do that stuff, we kind of branch off and do these consulting and these training things and we kind of stick everything in one single repo so when people do kind of go out and try to find someone like us, they can find us and kind of see what we're doing. It's been a really nice way to kind of get a pipeline of leads. We're not trying to make a billion dollars, and I don't think a lot of these other companies are either. It kind of makes sense in our space. If you're trying to get venture capital and kind of do something, yeah, you're going to have to probably take a different approach.

[0:43:17.0] JM: Yeah, content marketing as a way to build a personal brand or to build a technical brand, great strategy. That's kind of what I have been doing with Software Engineering Daily. I knew I want to start a software company and the road to doing that was unclear, and so I kind of started an intermediate, which was Software Engineering Daily which is not a venture backable business. It's not exactly a scalable business. It doesn't have the same scalable properties as a SAS product or some mobile app that I can just sell for a dollar.

In some ways, the non-defensible business model where you're doing something that's kind of commoditized, like training, or consulting, or making content. In some ways that's an easier business to get started with and to get profitable with because despite the fact it's commoditized, but there's so much demand for it that it's a really good business to just stand up and get going, and then you get yourself embedded in the community and there's all kinds are rocket ships that are being built in the community.

I think like — I don't really know where we were going with this topic, but if there are people who are looking for a business to start, you could do worse than starting with some sort of either education, or content, or consulting business. That doesn't seem scalable, but it can get you in a

position to where you're the master of your own destiny and you've got a positive cash flow of business.

[0:44:49.5] ND: Yeah, that totally makes sense. It's kind of like starting slowly and kind of getting yourself rooted in the community, like you said. Also, with something like your podcast, you've met everyone that kind of is active in all these different spaces. Now, if you get an interest in something, you have connections, you can kind of maybe reach out to these people and ask them questions, but you also have a very large breadth of knowledge about what's going on. If you listen to this podcast, they'll have a similar breadth of knowledge about what's going on in general, but kind of just knowing all of these stuff, it makes you kind of aware of what's going on. If you do have an idea, you can kind of pursue that idea much easier than if you kind of were not really aware of what's going on.

[0:45:31.3] JM: Yeah. Are you involved in the React Native open-source community? Do you contribute code to React Native open-source, or do you just write mostly?

[0:45:40.3] ND: My partner is Grabowski, and he's more of a contributor to React Native, the actual project. I've contributed a very small handful of times. Mike is actually in charge of releasing React Native to the community. Every time there's a new release, he actually is the one that releases that for Facebook. Him and his company; CallStack, which is his other company, they do all the release notes. Callstack is also another agency that does React Native and React consulting. By the way, they're similar to kind of Infinite Red and Formidable there, they're in Poland. They do a lot of work all across the world. They're very specialized in React and React Native.

He's contributed quite a bit to React Native, but also React Navigation, which is the navigation library. I've contributed more to — I'm part of the React Native community open-source repo. I'm also part of the, of course, React Native Training repo. We have a lot of stuff there. I've also helped with Reason React App, which is kind of like a way to build React apps using the Reason language I guess you would say. Yeah, it's kind of just whatever I feel like I'm able to contribute to, I will contribute to it. It's kind of different at a given point in time.

[0:46:53.6] JM: I had one specific question around building apps or just this certain situation that a startup might be in. Let's say a startup hires you to do training for their company because they have an iOS app that they want to get into android and the roadmap that they're suggesting to you is, "We're going to migrate our iOS app to React Native as much as possible as we build our android apps so that we can do a lot of code sharing in that process."

Is that a typical strategy the companies take? Is that a good idea where — Because I know it's commonly accepted wisdom that if you're building a mobile app, you start with iOS because that's where a lot of the early adopters are. If you want to get into android, maybe you — Actually, today the best practice might be kind of do this simultaneous rewrite of React Native in the iOS app while you're also building your android apps so that you can do some code sharing. Is that like boiling the ocean or is that actually a good idea?

[0:48:00.6] ND: You're saying, basically, like you have an iOS app, it's working great, you have users, and now you want build android. You're saying the best practice now might be to kind of start building android and porting over the iOS stuff and you're asking maybe what that would look like in a React Native world.

[0:48:16.9] JM: Yeah. I guess I should have phrased this better. I should have phrased it in a more open-ended sense, but more like if I've got an iOS app and I want to get my android app going and I also want to get on board with React Native at the same time, is that too much to ask? Is there a better route to doing that? What's the best route to getting to the glorious place where I can do code sharing and reuse between my React Native app and my iOS app and my android app?

[0:48:45.6] ND: Yeah. Right now, the way I would kind of tell a company to do that, and it would be my opinion, the best way. If it made sense based on their existing app, if it was something that was right for React Native, would be basically to kind of start rewriting cross-platform and React Native. Instead of just building for android, just go ahead and build for both android and iOS because you're going to be almost as efficient, if not, a little more efficient building with React Native anyway.

Kind of go ahead and port over everything into React Native both iOS and android and go ahead and kind of rebuild that using React Native and then you're able to kind of launch again using both platforms. That would be kind of the way I would say to do something like that.

[0:49:29.4] JM: Are there a lot of companies that come to you and they're in a situation where they've got either an iOS app or an android app and they're trying to figure out the best route to getting to both of those being managed with React Native?

[0:49:41.3] ND: That's not an enterprise company. That's almost always the case actually. A lot of the companies I work with that are not enterprise are kind of in that scenario, but instead of asking me of React Native is kind of the right choice, they've kind of already done research and they've come to the conclusion that this is probably the right technology for them. We'll usually consult with them before we go in there to just make sure 100% that this is going to make sense for them.

When it wouldn't make sense, I guess, would be a good thing to talk about. Like I said, games and stuff like that would be not a good use of the technology. Say you have a handful of really talented android and iOS engineers and maybe there's politics involved where they do not want to kind of learn React Native and it's going to kind of throw a wrench into your basic culture or whatever you have going on in your company, that might not make a lot of sense.

If you already have really talented developers and you're kind of iterating quickly and they're just might not — And you're okay with your payroll, it might not make sense to kind of start working on learning something completely new. I would say, in general, that kind of would be the scenario I would kind of throw out there.

[0:50:52.6] JM: Do you spend a lot of time training the other Facebook technology, the open Facebook technologies, like GraphQL?

[0:51:00.9] ND: No, I'm very specialized. I'm very specialized right now with React and React Native. I understand GraphQL. I've used it quite a bit, but it's kind of not something that I teach. I feel like the more — At least, in my experience, the more specialized you are, the better you

can kind of be at that one thing and just do it really well and kind of just know everything there is to know about that one thing.

I'm super interested in kind of expanding my breadth of knowledge, honestly. I feel like the more that I'll learn about React Native, the more there is to learn and the more I go meet with companies. The more questions that are asked that I don't have the answers to. I have to go back and learn and find those answers. Then I feel like, yeah, there's just always this rabbit hole of things that you want to learn.

Just based in my experience, the more specialized I've become, the better business has been. I feel like as GraphQL becomes more intertwined with React Native in the ecosystem, it will be something I need to kind of learn better and be able to teach. Right now, we teach pretty much strictly JavaScript and React Native. We don't even really go into any React stuff other than kind of teaching React to onboard people. We don't really take React Web clients.

[0:52:13.6] JM: To conclude, I've asked some people, who people who are deeply embedded in the React ecosystem. I always like to ask them; under what circumstances would it make sense for Facebook to build an operating system? I know this is totally speculative questioning, but I find it interesting because you see Facebook building all these interesting tools and you see React Native as this unifying force for mobile development. Can you see a world in which it would make sense for Facebook to build a mobile operating system or do you think this would maybe make more sense once we get to the AR world with the AR glasses?

[0:52:54.9] ND: The closest thing that I would say to keep an eye on regarding this question, because it's super interesting and I've thought quite a bit about it myself, is kind of what Expo is doing. I wouldn't be surprised if Expo became kind of that new distribution platform for apps. The problem right now we're seeing, of course, everybody kind of understands the issues with submitting two separate platforms to the play store and the app store and how that whole process goes and how it's kind of difficult for, I would say, new developers that don't really understand how to do that to keep going with it. Then if you have an update on the native side, it's kind of a process. Then with web, you're limited to kind of the DOM and the smaller bundle sizes and it's great and it's become way better than it used to be, but there are still limitations there.

I think what Expo seems to be doing is kind of creating that new platform and that new distribution system, and I would not be surprised if not Expo, something like Expo, would be that next thing, because with Expo you can kind of create and distribute very easily and seamlessly and kind of share things. It's a lot more open and a lot less friction. It's very very nice to work with.

I think something like Expo, I would not be surprised if a company like Facebook acquired Expo and kind of did what you just said. If it's not Expo, I believe it will be something that is along the same lines of Expo where they have innovated in this new way, and that's kind of something I would keep an eye on.

[0:54:21.6] JM: Yeah, but it sounds like you think something like this is going to happen. Whether it's in this current world that we're living in or we have to wait till there's a big platform shift to glasses or something.

[0:54:34.1] ND: Yeah, I believe so. I'm super interested in AR and VR and I love the idea of glasses and even the idea of contact lenses even, that kind of portray this virtual world on top of our existing world. It's awesome. I think phones are just so intertwined in our lives and we have — They're just very easily used and useful, you stick them in your pocket and you pull it out and you're kind of used to doing it. I feel like there will have to a 10X increase in productivity and just ease of use and just user experience for us to kind of move to that next platform. I do believe that will happen, I just don't know what it is. I don't think it's anything we've seen yet. I think it's going to be kind of a new thing that we didn't know even know about.

For that to happen, I believe it's going to be a few years longer than what a lot of people are expecting. When it does come, it's going to completely rewrite how we think about apps in general, really.

[0:55:29.8] JM: Can you imagine all of the work that has basically been wasted over the years because people had to write their apps in two places, and we just take that for granted today. That's just the way that it has to be done. I really hope that the next platform, whatever the hell it is, does not have the same rewrite problem.

[0:55:51.3] ND: I agree. I think we're going to look back on these days and we're going to just wonder what the hell we were thinking. We're limited to what we have, and it's getting better. There will be something that comes along and replaces React Native. There will be something that comes along and replaces the phone as a platform. I'm just very interested to see what that is and I'm excited about that happening, honestly.

[0:56:13.3] JM: Yeah. All right, Nadar, thanks for coming on the show. Everybody who's listening who is interested in React Native, check out React Native Radio, it's a podcast that Nadar hosts. Really interesting, a lot of topics around React Native. Thanks for coming on the show, Nadar.

[0:56:27.4] ND: Thanks for having me on, Jeff. I really enjoyed it.

[0:56:29.8] JM: Great.

[END OF INTERVIEW]

[0:56:32.8] JM: Thanks to Symphono for sponsoring Software Engineering Daily. Symphono is a custom engineering shop where senior engineers tackle big tech challenges while learning from each other. Check it out at symphono.com/sedaily.

Thanks again to Symphono for being a sponsor of Software Engineering Daily for almost a year now. Your continued support allows us to deliver this content to the listeners on a regular basis.

[END]