EPISODE 328

[INTRODUCTION]

[00:00:00] JM: React Native has unlocked native mobile development to web engineers who may now apply their skills to build iOS and Android applications in JavaScript. For the first time, cross-platform JavaScript-based applications feel as if they were written in the language of choice for the platforms. Businesses who choose to adopt React Native for their native app development also see great benefits, such as the ability to push new JavaScript code without going through the app store review process and the ability to share code and business behaviors across the iOS and Android platforms.

Expo is building a cross-platform, native runtime for React Native. Expo brings the benefits of deployment and iterative development to native without sacrificing any user experience costs. Expo plans to do this with their native SDK, their custom development environment, and tools built in collaboration with Facebook like Create React Native App, which is discussed in today's episode. React Native has the incredible potential to revolutionize all user interface development with its core set of cross-platform UI primitives and React's popular declarative rendering pattern.

So in this episode, Brent Vatne and Adam Perry join Caleb Meredith to discuss Expo and the future of React Native, and try to answer the question; can React Native become the one UI framework to rule them all?

[SPONSOR MESSAGE]

[00:01:39] JM: Every user request is unique. Different requests come from different geographic locations. Some requests should hit a cache, or a CDN, and conditions change overtime. Your application might get a spike in traffic. You spin up new servers automatically, and then the traffic dies down, and you spin down the server.

Modern application infrastructure is changing all the time, and your routing is changing all the time. Dyn provides DNS that is as dynamic as your application. Dyn's intelligent routing gets your users to the right cloud service, the right CDN, or the right datacenter. With 10 years of experience, Dyn is trusted by Netflix, Twitter, and Salesforce, to maintain uptime even in the face of unreliable networks and malicious DDoS attacks.

Dyn is a DNS solution for products and companies of all sizes. Get started with a free 30-day trial for your application by going to dyn.com/sedaily. After the free trial, Dyn’s plans start at just $7 a month for world-class DNS. Manage the internet like you own it. Go to dyn.com/sedaily to get your free trial of Dyn DNS.

[INTERVIEW]

[0:03:10.3] CM: I’m here with Brent Vatne and Adam Perry who work on Expo, a framework that helps JavaScript developers build native iOS in android applications. Brent and Adam, welcome to Software Engineering Daily.

[0:03:20.8] AP: Thanks for having us.

[0:03:22.4] CM: All right. First, I want to talk about Expo. We have had many cross-platform tools that allow developers to write web apps that can be used as a mobile app. Is the application experience of a web app so different than a mobile app that there is a tension that cannot be resolved in some of these previous cross-platform tools?

[0:03:42.3] BV: I can take that one. I think that there are a few things that maybe need to be discussed about the differences between some of the other solutions that are available for the same problem and whether or not the approach of taking web applications and making them into mobile applications is a viable strategy.

I think that some of the other solutions that have existed before have been, in many ways, trying out ideas and approving concepts that ultimately led to React Native. This is something like Expo as a result of that. Things like Titanium, which as far as I know were the first ones to pioneer the idea of having some app code running in JavaScript and sending any updates over the bridge in order to update some native UI.

This was the concept that existed well before React Native itself. Even through Cordova or Phonegap, there was a way to communicate over the Cordova Bridge, which was admittedly more simplistic but the same idea where you have some app code running in this language that's shared across platforms that then sends some commands over to a native side in order to either call into APIs such as open a camera or even APIs create a view and position it here on the screen.

This is something that, I think, it's just grown over time. As for whether or not web applications can or fundamentally at odds with the idea of creating a mobile application, I don't think that is necessarily true. I just think that with the current state of tools that exist for the web, it's very difficult. You hit a ceiling at some point where you just can't take the UI any further. You can't handle gestures any better. You can't do it in a cross-platform way due to a lot of discrepancies around how different browsers handle certain events or support or don't support different things.

It's less of an idea of it not being theoretically possible and more. I think that it's just not practically possible right now, if that makes sense.

[0:05:57.7] CM: Yeah, of course it does. React Native is an incredible tool that solves a lot of these problems and ultimately empowers web developers to build applications for iOS and Android that feel as if they were written in the native language for the platform because of bridge technologies. What aspects of React Native make it difficult to use and therefore warrant Expo's existence?

[0:06:21.9] BV: Adam, you want to take this one?

[0:06:23.6] AP: Yeah. I think there are a number of categories of problems that people run into when using React Native on its own without using Expo's tools. Most of them are in some way related to the effort required to configure, build, and manage updates to the native code in a React Native project.

When you compare the iteration cycle of React Native to the web, once you gotten up and running with React Native, it tends to be quite strong but there are a number of headaches associated with getting to that point where you can be iterating quickly on your UI. Somebody thinks they're just initial startup problems, xCode in Android studio, especially for beginners, are really difficult to wrangle.

Then even for experts, there are ongoing issues with tracking the stability of React Native's native header files and APIs for accessing the bridge. Keeping whatever set of open source native modules you're relying on up-to-date and in sync with your project can be quite a challenge for a lot of teams.

Finally, writing your own native modules to begin with is it's not nearly as well-documented right now at least compared to the JavaScript APIs. The training, the tooling, and the culture around React Native are very much focused on working in JavaScript, and so it can be very difficult when developers want some native primitive available to them but don't have the expertise or bandwidth or resources to dropdown to that native code and build their own native APIs.

A number of these problems are ones that we attempt to solve at Expo. We provide a curated set of native modules that we're growing that list and trying to offer the primitives that people can work with in JavaScript so they don't have to dropdown to native code. We also handle all of the native build and configuration as part of the Expo client app that loads your JavaScript directly, so you don't have to manage or configure or manage updates to the native part of your React Native Application.

[0:08:34.5] CM: What are some of the native modules Expo provides to its users?

[0:08:38.4] AP: We have a whole lot actually. They range from more or less what you'd expect like an image picker API that's cross-platform. I should mention also that cross-platform APIs is something we prioritize so that developers don't often have to worry about the platform differences.

We also have GPS and video, and we're building on audio API right now. We also have some slightly more exotic things. We have a very interesting end novel OpenGL implementation that's compatible with WebGL. You can use three JS inside of Expo, and it works as it would on the web. It's fundamentally operating directly on the native OpenGL APIs. Then, we have a whole list on the Expo documentation.

[0:09:19.6] CM: I think a few more worth highlighting there, because just going back to the original explanation. One of the key things about Expo, I think, is that it just makes you productive out of the box as soon as possible. Whereas, when you're just using React Native on its own, there's a lot of overhead for learning how to configure an ex code or android project for how to get it somewhere where you can actually share with your teammates or other stakeholders, all sorts of things like that.

Part of this is, we've identified, there are things that people just need from most apps. Some of these are Facebook login, for example, or Google login, and then we also just provide a generic wrapper around some platform-specific model web browsers. Going a little bit into detail about the APIs, it's called SFSafariViewController on iOS and Custom Chrome Tabs on Android. What that does is it shares cookies with your actual browser app so that you're able to easily implement OAuth and things like maps, which are very common.

These are all sorts of things that you can, for the most part, just install in an existing or rather a new React Native App that doesn't use Expo, but it just adds on all of this work involved with actually figuring out which one you should use out of these libraries that exist in the ecosystem, keeping it up-to-date with everything. The various kinds of configuration that are involved with that, that it turns out, if you're coming from a web background, it can be pretty overwhelming when you need to start up a new project and add all these things in and learn so much at once. We're just hoping to make the experience ultimately a lot more web development, where you just create an app and load it into this client and start working on it and you have the APIs that you need to build your app available to you right away. That, I think, summarizes it pretty well.

[0:11:09.7] BV: Yeah, I think one other piece in addition to the develop tooling, the native modules that I would add is that we're building up a number of services to make it significantly easier for people to leverage actual native-specific technology. We're not shooting for parity with web applications, but we have a push notification service where you interact through JavaScript with our API, or with the notifications that are coming on the device, rather, and you have full control there, and we also have build services so that when you want to publish to the app and play stores, you just — Just a few commands on your command line and you have an IPA file to download and upload to iTunes Connect. Which really streamlines across us for a lot of people.

[0:11:54.8] AP: Yeah, just one follow up to that. In case we missed anything here, you can check it out on Slack Overflow. We're answered this question and it's probably phrased more eloquently than what we've managed to put together on this box. Yeah, check that out.

[0:12:11.7] CM: Yeah. I think you explained it quite well. Having those primitives are especially useful for JavaScript developers who are not familiar with the web platform. Saying that, one of the real powers of React Native is that even though it allows you to primarily write your app in JavaScript is that when the JavaScript is perhaps too slow or you need a native API that Expo or React Native doesn't provide, you can jump down to that lower level and write a native module. Expo provides a bunch of native modules on its own, but is also discourages users from writing their own native modules or bringing in custom-native modules from the NPM ecosystem that aren't available in Expo.

Why would Expo discourage this functionality?

[0:12:55.7] BV: I think the main reason for discouraging it is just that it limits a lot of the services that can be provided around Expo. Things like handling your builds for you or having an easy way to, immediately, as soon as you create the app, share the URL with somebody on the other side of the world so that they can load your app up and live reload as it's happening.

These sorts of things are only possible when you have a shared native run time. As we've seen with the web, there are quite a few things that you can do once you have the shared native runtime essentially, which is really what the web APIs are. We discourage that because we think that there are a lot of benefits to having that.

That said, I think in practice, there are cases where, especially as React Native is still two years old or so and Expo around the same age, you're going to two cases where you kind of will need to, in some cases, bridge a native API. What we've done is we made it possible for you to either call it eject, kind of the in the context of Create React app, or Create React Native App, or we like to call detach within the context of Expo itself. Feel free to read the documentation for more information on why that is.

What it does, essentially, is it generates iOS and Android projects for you where, normally, with Expo, you just have JavaScript files. When you run this command, it generates the iOS and Android projects and links the Expo native SDK, so that you're then free to add whatever native code that you like to the project. Of course, you lose out on a lot of the benefits of having the shared native runtime, but if it's absolutely required, then it's definitely a possibility.

[0:14:42.0] CM: Yeah. Shared native runtime is a great way of describing it. Do you think Expo is the best choice for building greenfield React Native Apps, or will large apps inevitably eventually bring in their own custom native modules and eventually ditch Expo with?

[0:14:58.5] BV: I think that if you know right out of the box that you're going to need some custom native module, like maybe you're making a video or chat kind of app or you're planning on using Bluetooth. Someday, we plan on having support for those APIs. At the moment, you kind of just have to look at what's available through, as we call it, Expo Kit, which is this native SDK that you get when you detach. Say, whether that includes other things that you need, or if it's worth just starting from scratch with a plain React Native project.

I think many cases, there is no downside to just starting out with Expo. If you hit a wall, you can just detach and continue on as if you had started from nothing before with a plain React Native project.

[0:15:46.8] AP: That's a pack that I would probably plug myself. I think there's — At this point with Expo Kit maturing, there's very, very little downside to doing some initial UI prototyping and like starting work with an Expo just because it's so fast to get running and you don't have to worry about a number of new project set of elements. There's always the escape patch.

[0:16:12.5] BV: Yeah. It's important to emphasize as well that we don't lock you in in any way, you're free to take your code anywhere or use our native SDKs. It's all open-source. By starting off, there's not really — Like Adam said, there's not really much the downside to doing it that way, and there's a lot of upside and that you can just get going without having to think about a lot of these details that maybe you don't even need to think about yet on your project.

[0:16:36.3] CM: Okay.

[SPONSOR MESSAGE]

[00:16:44] JM: Every software project uses email. Every time an eCommerce site processes a transaction or a user makes a comment on a social network, email notifications are sent. SparkPost provides email delivery services for apps and websites. To try SparPost and send 100,000 emails a month for free, go to pages.sparkpost.com/sedaily.

SparkPost has a range of pricing options from free self-service packages to sophisticated enterprise support and services. Start sending emails to your users today. Go to pages.sparkpost.com/sedaily to send 100,000 emails a month for free.

Thanks to SparkPost for being anew sponsor of Software Engineering Daily. If you want to send 100,000 emails a month for free, go to pages.sparkpost.com/sedaily.

[INTERVIEW CONTINUED]

[0:17:51.3] CM: Next, I want to talk a bit about Create React Native App. At React Conf 2017, Expo announced a new project; Create React Native App. What was the need you saw in the React Native ecosystem, and I guess, in the Expo ecosystem, that led you to build Create React Native App?

[0:18:09.0] BV: Yeah, we had some conversations with some folks at Facebook who work on React Native open-source and kind of the products of these conversations was that it would be really fantastic if there was a way to make use of a number of Expo's high points that the qualities that make developing on top of Expo really wonderful without having to depend directly on our backend services.

Our services offer a number of very useful things to developers, but when it comes to getting start to experience for a technology, there was definitely a gap for something that had the ease of use of the Expo client without having to go through an account registration and reading a lot of documentation. There's also in the React Community at large, on the web and on native, then a lot of praise heaped on Create React App. It's provided an excellent model for a CLI for React

Project, something that React projects were the community hadn't really rallied or on the standard standard. There were some attempts at doing it well, but there was not anything that was kind of the de facto tool to use and Create React App, as far as I can tell, has become that default.

It's a very familiar model to developers who want to work with React, and there's also some nice benefits as an implementer of a tool when you're doing it the way that Create React App does, but that's kind of getting into the weeds a little.

Ultimately, we identified a lot of work that would necessary to make Expo play well with no service dependency and with the Vanilla JavaScript that runs React Native normally. Expo projects, it's very slim. There's not very much different in it. Historically, we've used a fork to make sure that we can offer the best experience possible. There was a lot of work that went into these elements to make sure that Create React Native App was a polished way for people to get started with React Native in general without having to also, at the same time, take the plunge on buying into Expo's model.

[0:20:18.2] AP: Yeah, I think it's really important to emphasize from that. The getting started process for React Native without using Create Native App can take up to hours depending on your internet connection and what you have installed already. You have to install xCode and the command line tools and various other things.

A metric that was really important for some of these open-source folks at Facebook were considering was what is the time to Hello world? Before Create React Native App, the time to Hello world, say, could be hours, which is too damn long, frankly.

With Create React Native App, one of the main objectives was we want to get this startup time, this time to Hello world, down to the minimal amount of time that we can possibly do. This is an ongoing effort as we're improving various tools and removing dependencies. At this point, it's pretty straightforward. If you have an iPhone and you have a Windows PC, you can install a couple of programs and essentially get started within 5 or 10 minutes, if not less. It's a significant improvement and it opens the door to development for devices like iPhones, on

Linux, and Windows where that wasn't previously possible using the model suggested in the React Native documentation.

[0:21:45.5] CM: Thank you for providing context.

[0:21:48.7] AP: I'm in too deep.

[0:21:51.7] CM: Just to provide context really quickly to our audience, we're talking about Create React App and Crate React Native App. Create React App is a web version, and Create React Native App for iOS and Android. Yeah, the names are similar, it can be a little confusing. I just want to make sure that clarification is out there.

What I'm hearing from you is Create React Native App was Facebook's idea?

[0:22:13.4] AP: I wouldn't say it was entirely their idea. I think it's difficult to attribute that when it's the product of kind of an ongoing conversation. I've been the main committer on the tool itself, but we've been working on the Expo side on a bunch of things, and we've had help from Eric Vicenti and Martin Konicek on making sure that upstream thing happen to provide a smooth process to transition from Create React Native App to using the full React Native CLI. It's definitely been a good collaboration, I think.

[0:22:47.4] CM: Okay. The name of Create React Native App as we discussed previously was inspired by Create React App. What is Create React App and how is Create React Native App similar? If you could go into more detail in that.

[0:23:04.2] AP: Definitely. Create React App is a command line tool for — Strictly speaking, Create React App, it's a command line tool for initializing React projects, and it does this using this other NPM package called React Scripts. Basically, the way that works is Create React App itself is completely stable. Users don't have to upgrade the global command line tool, and this stable global command line tool downloads an up-to-date scripts package.

A scripts package contains all of the build and tooling configuration for this default React project. That includes CSS and JavaScript bundling and includes Babel configurations. It also includes

Jest testing config, so users have a testing config out of a box. It includes a fantastic user guide on how to extend your React application with things like Flow type analysis and other things you might want to include in your project.

This model for Create React App, it enables a few things. I think if you talked to the maintainers and creators of Create React App, they would have a number of other positives. Basically, it means that users don't have to update their tooling globally on a regular basis, and so tooling in configured per project. It also essentially has the effect of hiding complexity from the user until they're prepared to face web part configuration on their own. It bundles all of these best practices for React applications under this script package that it installs for you. We have a very similar model for Create React Native App. This is not something new to the world. We saw it working very, very well for Create React App and basically just went to the same route.

We have a React Native scripts package, and this includes a number of tools both from React Native and some stuff we built at Expo to smooth out a few things that enables users to run all of their build commands for their project from the configuration. That's just this one single dependency that they have in their project.

We're not quite as well abstracted as Create React App yet. There are a couple of points where we don't have programmatic control over certain things like Babel configuration the way that you do with web pack. There is ongoing work to fully encapsulate all of that build and tooling configuration, so we're not there yet, but we've managed to get, I think, the bulk of what a user would normally be confronted with if configuring all of these themselves and essentially hide it in this package. They just have to update a couple of developer dependencies in order to keep their build tooling current with the ecosystem.

[0:25:51.8] CM: It's important to stress that Create React App was incredibly important to the React web ecosystem. Before it create React App, as you mentioned, everything that Create React App does, you'd had to do manually before you could even set up projects. This would lead to hours of frustration before you could even start hacking on that.

[0:26:10.7] AP: Absolutely.

[0:26:11.0] CM: Create React App made starting a React Web app much, much simpler. Is Create React Native App saving as much frustration for React Native developers, especially when we consider that the cost of signing up a React Native project is mostly a one-time cost to install xCode and Android Studio? After that, starting a React Native project is pretty easy. Is Create React Native App saving us much frustration?

[0:26:34.8] AP: That depends on which developer you talk to. You have the right to identify that the React Native CLI has definitely done a lot of the work for React Native that Create React App did for React dump, or React for the web.
Create React Native App is — I think from one perspective, if you're an existing mobile developer, Create React Native App is kind of closing that last mile. I think there is another group of developers that we would really love to see in the React Native ecosystem for whom the native dependencies were just a deal breaker for trying things out.

That's existing web developers and people new to software development. Basically, anyone who hasn't dealt with xCode or anyone who owns an iPhone but not a Mac, because for React Native CLI project, you have to have a Mac if you want to develop for iOS. I think it's important not to downplay the existing amount of work that went into React Native CLI and its template projects and its scripts.

I think you're absolutely right to identify how easy that is to do once you've gotten all of the build configuration startup. I also think that these people who just were not a part of the community before and who might have a chance to get involved and at least try things out, it is big.

I also think that, and there's been some work elsewhere on this problem, but upgrading React Native versions has historically been challenging. There's been work on this React-native-git-upgrade tool, which I have not played as much as I would like to and as much as I probably should have, but it aims to essentially use Git Merge technology to help with upgrading your React Native version, especially when it comes to your custom native dependencies.

This is something that is, for the most part, abstracted away for you when you're using Create React Native App and by extension when you're using Expo. I think that that's also a significant boon to existing React Native developers if they were to try it out.

[0:28:42.7] CM: Yeah, certainly. Being able to develop iOS on Windows is definitely a key seller there. You might have mentioned this already, but being able to develop iOS on Windows or Linux, is that excusive to Create React Native App, or does Vanilla Expo provide that functionality as well?

[0:29:04.4] AP: Yeah. It's possible in Create React Native App, because we load your project in the Expo client. This is enabled by the same underlying technology which enables cross-platform development on the Expo tool kit.

[0:29:20.1] CM: Could you develop iOS on Windows with just the Expo tool kit?

[0:29:25.1] AP: Absolutely. Sorry, I didn't answer your question directly enough. I apologize.

[0:29:28.2] CM: Okay. All right.

[0:29:29.1] BV: I think another thing that is worth pointing out about this is although currently, the only way to open a Create React Native App project on your phone is with the Expo client, there's no reason why it has to remain that way. Anybody can really build their own client. They can open a Create React Native project. It would be really interesting to see their people work on that and see what we will come up with.

There's an up chase on file, which defines the configuration for various things like what icon to show on the loading screen or various things related to, let's say, push notifications and whatnot. This is all under the name space currently of Expo, so if you open up JSON, you'll see curly braces and then at the top level, there will be an Expo key.

The configuration that lives under that is specific to Expo, but you could imagine there being another key. Maybe Software Engineering Daily, and then you have your own client for Create React Native App. There's nothing really stopping that from being the case. It would be really interesting to see that happen.

[0:30:31.5] CM: That would be fun. We could download episodes in the background or something.

Next, I want to talk a bit about React Native on other platforms. React Native is not limited to just iOS and android development but may theoretically target any platform where JavaScript may run. Some other platforms built for React Native today include one for the Windows phone virtual reality and, ironically, the web. Do you foresee React Native becoming a popular framework for building on other platforms than iOS and android?

[0:31:03.9] BV: It's worth pointing out that even the Windows implementation is not just for Windows phone, but it's for the universal Windows platform. It runs on HoloLens. It runs on Xboxes or desktop. It's pretty much anything that runs, I believe it's from Windows 10 or higher. The API is considered universal Windows platform. Someone, I'm sure, will correct me on that, which is fine.

To go back to the original question, yeah, I think we're seeing this, and we have been seeing it for a couple of years now. It doesn't necessarily have to be a mobile device or some new medium. It can be something that we've been using for years in a different approach on how to rate a UI for it. At React Conf this year, someone mentioned, I think it was Dustan Kasten, React-blessed, so you can write terminal UIs using React.

They even use React-motion to create animations in that context, which was pretty interesting. There are other things. I think Dustin also built React-hardware, so you can use React to control Adruinos and that type of thing.

Yeah, I think, ultimately, that what React has going for in this space is that the UI that you define or rather the virtual dom that you define through JSX really isn't tied to any underlying implementation, and you can quite easily implement these renders that do completely different things than what you do in the context of a web browser or a native app.

As far as I can tell, one of the more interesting approaches to having kind of a cross-platform configuration and declarative, I guess, in some cases, logic or UI, it's out there. Ken Wheeler has also done some pretty interesting stuff around that. I believe he's going to give a talk soon about powering a crossbow with React. I have no idea what that means, but I guess we'll find out.

[0:32:56.7] CM: Yeah, I guess we will.

[SPONSOR MESSAGE]

[00:33:05] 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 Incapsula. To stop attackers and improve performance. When a listener makes a request to play an episode of Software Engineering Daily. Incapsula checks that request before it reaches our servers and filters bot traffic, preventing it from ever reaching us. Botnets and DDoS attacks are not just a threat to podcasts.

They can impact your application too. Incapusla can protect your API servers and your micro services from responding to unwanted requests. To try Incapsula for yourself, go to incapsula.com/sedaily and get a month of Incapsula for free. Incapsula's API gives you control over the security and performance of your application.

Whether you have a complex micro services architecture or a WordPress site like Software Engineering Daily. Incapsula has a global network of over 30 datacenters that optimize routing and casher content. The same network of datacenters that is filtering your content for attackers is operating as a CDN and speeding up your application.

To Try Incapsula today, go to incapsula.com/sedaily and check it out. Thanks again Incapsula .

[INTERVIEW CONTINUED]

[0:34:51.7] CM: React Native has become, I guess, you could say a popular framework on iOS and Android. Do you think it could also become popular on something like the universal Windows platform?

[0:35:01.8] BV: Yeah, I think so. I think that the label of React Native is probably going to go away. I think it will just be React and there will be sort of React iOS and React Android and whatnot. There's been a lot of work and I think that the universal Windows platform implementation of React Native is not only maturing relatively quickly and gaining kind of mainstream acceptance within Microsoft itself, which we saw by having the repository being moved from the React Windows organization to the official Microsoft organization on GitHub recently.

There's a lot of work going on to try and keep parity with the iOS and Android versions of React Native, which is no small task considering that there is a much smaller group of people who are working on the EWP React Native platform.

Yeah, I really think that as long as work continuous on that, there — It's kind of a limiting factor and that most people use React Native specifically for mobile UIs at the moment or mobile apps. Of course, Windows phone isn't quite as popular as maybe it once was or could be. Yeah, it really depends on what people start to use it for. I could absolutely see that becoming more mainstream.

[0:36:20.1] CM: You mentioned that you thought, maybe eventually, the native label would just disappear and just become React. What happens to what we classically think about React, the JavaScript library that does the reconciliation between the virtual Dom trees? What happens to that library of React Native becomes React?

[0:36:40.6] BV: I think it's pretty similar to what's already been happening over the last couple of years where React Dom itself was pulled out of the core of React library and I think there might still be some vestiges of React Dom itself in the core. I think we just kind of see these different renderers becoming their own packages that using conjunction with the core library that's responsible for doing this underlying reconciliation and whatnot.

[0:37:10.7] CM: Do you think Create React App will eventually React Native under the hood?

[0:37:18.3] BV: The future in which that happens is so far off that it's very difficult to predict. I think there's about five million sequential events that would have to happen before that. Now, I'm not sure about that. Whether it will happen or there's desirable, I think it would be amazing if that were to happen. I would love to see moving mergence between React Dom and what we currently call React Native. I think that that would be incredible.

There are a lot of opportunities for that Nicholas Gallagher at Twitter has done a lot of work on React Native Web and is continuing to push forward with that. Lillian Richardson who works at Airbnb has similar library called React Primitives, which is pushing for that.

Ultimately, the — I was saying earlier that the platform is really — There's no reason why, I guess, theoretically speaking, that web applications cannot also be mobile applications. As we see more convergence there, I think it will make this a lot easier.

Also, as React Native matures and in many ways is becoming more spec compliant with the web APIs that uses, for example, with Flexbox, which is pretty close now. It makes this kind of thing a lot easier to do. I've heard of people who are working on apps internally at Microsoft who, one of the developers there, actually rewrote the implementation of CSS layout, which is now known as Yoga, in order to get better spec compliance so that they could reuse code from iOS and Android that involves using Flexbox layout on the web as well.

Yeah, I think —

[0:38:49.2] CM: Wait. Sorry. Microsoft is using Yoga now? Yoga is the implementation of Flexbox and React Native.

[0:38:56.3] BV: Yeah. They actually re-implemented that also, I believe in C# for the universal Windows platform. Similarly, they're using React Native on a couple of projects internally. This was something — I believe it's been discussed publicly and you can actually try the private beta of it now. There is the Skype preview for Android which is build using React Native. Yeah, they're really experimenting with a lot of things related to React Native over at Microsoft. They have their services side which is working on the — What was the name of this thing? [inaudible 0:37:47.4] is going to kill me.

[0:39:35.2] AP: Azure Mobile center.

[0:39:36.3] BV: Mobile Center, that's the one. This is something that is kind of end-to-end integration where it builds analytics and error reporting and all of these kind of thing into your React Native project for you along with code push which gives you the over the air updates kind of out of the box. It's sort of — Sorry. I guess — Lost the thread of the original question. I guess, that's how it is sometimes.

[0:39:59.5] AP: Also, I would add to this general discussion that in the long term, there's a lot of interesting possibilities with a full-scale convergence of React Dom and React Native. In the near term, regardless of how the potential for that process work out, in the near term, there's a lot of very interesting work going into improving — The same way that React Native improves co-chairing across iOS and Android for mobile teams. There's a lot of very interesting work, some of which Brent mentioned on making it easier to abstract over the differences between Dom and the native UI layout so that you can share React code, and share UI components, and share business logic, and these things are becoming much more a reality in the near term, which I think is really exciting.

[0:40:43.6] CM: Just going back to the Yoga thing really quickly. I want to make sure I'm understanding this correctly. Microsoft is thinking about using a Flexbox implementation, originally written by Facebook in Edge when Facebook isn't even a browser company.

[0:40:58.2] BV: Oh, sorry. They're not using it in Edge. They're using it in the UWP React Native implementation.

[0:41:04.5] CM: All right. Okay. I'm sorry. That's just how I heard it.

[0:41:08.2] AP: I'm glad you clarified that.

[0:41:09.7] CM: Yeah. Okay. The React for virtual reality project, or React VR uses React Native. Why would React Native be an attractive choice for a greenfield project like React VR?

[0:41:22.5] BV: That's a good question that I don't think I have a good answer to, unfortunately.

[0:41:27.2] CM: That's okay. I'm just kind of thinking about it loosely, the build tooling, if nothing else.

[0:41:35.1] AP: I think a JavaScript bridge to native API is a powerful abstraction, and when you have a robust working on. I can't speak to the motivations for building it this way, but I can certainly sympathize with the desire to reuse that code. That seems powerful to me.

[0:41:53.2] CM: Okay. All right. React started as a UI framework that only targeted the web, and we talked about this a bit earlier. Now it has grown to support more platforms, React Native. React Native. Ironically, one of the platforms that React Native now supports is now the web, which as we know, some employees on Twitter are working on. Why is it attractive for React Native to support the web platform when React itself already does a good job at that?

[0:42:20.2] BV: I think that a big issue with the way that the interop works right now is — I think we're not necessarily talking about replacing the React Dom package, or anything like that. I think a key part is on React Native and on the web, or React Dom rather, there are different component that you use to describe primitives. On the web, you'll have a div, in React Native you'll have a view. Currently, if you import a view from React Native, you essentially can't use that component that imports the view. You can't share that with the web.

What React Native web and React primitives do is essentially give you this package that where you import these primitives from rather than importing them from React Native itself. They take care of mapping the properties that are common or that work on the web to the appropriate properties for a div and for spam and what not. It provides this layer that kind of translates between the way that React Native describes components and the way that React Dom describes components.

[0:43:25.2] CM: From my understanding, the mobile Twitter website uses React Native for web, and it's in the entire app. The Twitter iOS and Android app does not use React Native at least yet. Twitter is only React Native to target the web. Why would the mobile web Twitter team make the decision to user React Native for the web? I do want to emphasize, this is the mobile Twitter website and not the main Twitter website.

[0:43:51.3] BV: I think that's a fair question. One thing I'm not actually not entirely sure about is to what extent React Native web is used on the mobile Twitter site. I know that Nicholas has mentioned using React Native web and the way that styles are aggregated and de-duped and whatnot, and using that to build out style sheet. I'm not sure if that's what's being used in production or not. Maybe you have more information than me about that.

It's really hard to say. I think that Twitter already has a lot of mobile engineers and it's a big cultural shift to go from having native mobile engineers to now starting to integrate React Native into various parts of your app. Some companies are slower than others at doing it, because maybe they're more cautious, or have different philosophies around mobile development. We've seen Khan Academy has only recently started to look into integration React Native into their mobile app despite having a team that involves — For examples, Jared Forsyth who worked at Facebook before and worked on React Native and dev tools around that and was advocating for it for a while.

I think people just have — And organizations have a different tolerance and what stage of maturity for a project they're willing to start making big investment into it. That's seems to me to be likely, but this is all speculation. You should definitely have Nicholas Gallagher on the podcast at some point to discuss Twitter mobile. I think he'd a great guest.

[0:45:18.1] CM: Yeah, for sure. I guess my only reference point is I opened up the React dev tools once and I saw some views.

[0:45:25.7] BV: Yeah, they are using React on the mobile site. That's — Yeah.

[0:45:30.9] CM: Yeah. Do you have any other unusual or ironic examples of people using React Native?

[0:45:37.7] AP: I wouldn't classify it as extremely unusual, but there's been work going in to the tvOS target, I believe for using Apple TV. I think there are some other interesting, not targets that you don't immediately think of when you think of web technologies. A lot of — I think there's been work into — Before I misspeak, I'm going to stop. I can't remember what — There's been some work in the tvOS and other kind of kiosks-style technologies.

[0:46:03.8] BV: Yeah. Maybe this isn't a very outrageous example, or funny example, but more of a serious one. I was really surprised to learn that Flipkart which is a very large retailer in India, they actually use React Native on one of their mobile apps for Android, which is used primarily by people who have extremely low-end devices. They might have, at most, maybe 60 megabytes of RAM available when they open the app. I think React Native has a bit of a reputation for using a lot of memory, so they do all sorts of optimizations around that related to list views and whatnot.

I think what that really demonstrates is even in really harsh conditions, you can adapt React Native to work for your use case. It might take a bit of work, but that seems like a — It's been valuable, at least, in the context of Flipkart. That's another person who would be a great podcast guest, is Pjani. I'm sorry if I'm saying the last name wrong. He is a really smart guy about progressive web apps and React Native. Definitely chat with him further about that.

[0:47:10.8] CM: Okay. Does Expo ever plan to support platforms other iOS and Android?

[0:47:17.0] BV: I would love to support the web as a target for Expo. I would love to have progressive web apps being a first-class citizen and the Expo ecosystem.

[0:47:27.8] AP: I think we're constrained by engineering time, not by desire or imagination.

[0:47:33.3] BV: Universal Windows platform has been requested countless times at this point. That is just something where we're iterating really quickly on our iOS and Android clients, and so it's difficult to take the time to really throw a third one into the mix. Once we maybe get a few more engineers on board, we might revisit how what we're targeting.

[0:47:54.1] CM: Yeah, that sounds awesome. Given that we see React Native becoming popular on other platforms like iOS and Android and including some growth on the web, can we conclude that React Native is the best paradigm developers have today for creating user interfaces between declarative rendering and React Native's common UI primitives.

[0:48:16.8] AP: That's a bold statement.

[0:48:19.3] CM: I'm asking you to make the statement.

[0:48:22.2] BV: Sign on the dotted line. I like it a lot. I don't know if it's the best one. There's still — Different apps have different needs. For some applications, maybe React Native wouldn't be a great fit for the entire thing. I think there are a lot of cases where it does make sense. I would maybe go as far as to day that for a large portion of apps, React Native may be the best way to build apps. That might be the most extreme statement that you'll get out of me.

[0:48:54.3] CM: Okay.

[0:48:55.5] BV: I think that there are a lot of good options out there. Yeah, personal favorite, of course, being React Native. There are just some limitations currently to the Async model that React Native uses for communication across the bridge. Things that are related to, for example, gestures that if some of these issues were resolved, and there are ongoing efforts to try and come up with better APIs. If these issues were resolved, I would be confident in saying that it's definitely a contender for most apps as being the best

[0:49:28.0] AP: Taking a quick look through my phone home screens, I don't see anything which jumps out to me as like, "I couldn't build this in React Native," which I think isn't a pretty good endorsement. Granted that I don't play very many mobile games. I think that's an area where answers for — There's been some interesting work including by one of our teammates. I think there's a lot of promise there. But most app categories that I interact with on a day-to-day basis, I think React Native would be fantastic for them.

[0:49:55.0] CM: Even the game case. You could see web assembly maybe

[0:49:58.2] AP: Yeah. You can see using it, there's more interest these days in mixing React Native with existing native iOS and Android applications. I could definitely see for a game using

React Native for menus and for UI elements and using wasm and GL under the hood for the actual game. There's a lot of possibilities.

I think that a phrase that I — I don't remember where I heard this recently. It was at React Conf. Someone described it as a very high-ceiling technology, where you'd get a lot out of the box, but you can also take it a lot further in a typical cross-platform mobile toolkits. My opinion is that's a very apps description. People build some pretty impressive stuff with it, and I think it's very powerful when you combine the expressivity with the iteration time and the UI performance that you get from using the underlying native primitives.

[0:50:49.9] BV: Yeah. I think, Caleb, what you pointed out earlier about an advantage with React Native of always being possible to dropdown to native, is really important for this kind of discussion, because you're really not limited in what you can do with React Native. You can use it for a single screen in your app, some small piece if you find it useful there, or you can use it for your entire app and just bright wrappers for components that you implement natively as you need it.

Anytime you run up against potentially some limitation of, for example, gestures, or lists, you can absolutely just write a native wrap or a native view and drop that in. The flexibility is definitely something that makes it extremely valuable.

[0:51:34.5] CM: In the future, will React commonly be used without React Native on the web, or do you see a lot of React developers adopting React Native for their web apps?

[0:51:44.6] AP: I think, for that to happen, I would expect to see a lot more maturity in the compatibility layers that people are building. I think there's a tremendous amount of potential and promise there, but my opinion at least is that's a little early to make conclusive statements about how exactly that's going to play out. I think it's really exciting though. I think it definitely could be that way.

[0:52:03.9] CM: If we say, hypothetically, the integration is perfect and up to standards with iOS and Android, do you think the idea, at least, the concept is attractive?

[0:52:14.2] BV: I think so, yeah.

[0:52:15.5] AP: Yeah, I don't see why not. I'm not going to speak for other people who are picking their tools. I think having good abstractions for what a UI element is without binding directly to the language used by web browser could be very powerful.

[0:52:30.4] BV: Well said.

[0:52:31.9] CM: Given that Facebook has a lot of developer mindshare with React and, now, React Native, do you think it makes sense for them to build a React Native Facebook phone?

[0:52:44.6] BV: I think they've tried to build a phone before in the past, haven't they?

[0:52:47.6] AP: Have they? I don't know.

[0:52:48.5] BV: I guess who hasn't, really? It might make sense, are they going to do it? Probably not.

[0:52:56.3] CM: Why not? Why do you think?

[0:52:59.4] BV: Maybe years down the line. At this point, the way that I see it is the project is maturing and growing so rapidly that building a piece of hardware around it is probably not the greatest decision over this period of time. If the platform stabilizes, and I guess there is a desire for it at Facebook, then maybe — They have so many resources.

[0:53:23.0] AP: For Facebook specifically, they're not using React Native for the entirety of their mobile application. I think that — From an incentives perspective, I don't see them doing that anytime soon. It's a fun idea.

[0:53:36.2] BV: Are you planning on building a React Native phone, Caleb? Is that why you asked?

[0:53:39.0] CM: No.

[0:53:42.5] BV: Is that something?

[0:53:44.3] CM: I think that's something at least Christopher Chedeau has talked about in the past where it's the holy grail if you can the React reconciliation algorithm running as low as possible on the device. Yeah, it's an interesting —

[0:53:58.7] BV: Yeah. I think Cheng Lou's talk from React Conf last week was kind of related to that —

[0:54:05.8] CM: Very exciting stuff.

[0:54:07.6] BV: Yeah, you have these abstractions and you just try and push them deeper and deeper so that you can then start concentrating on higher level problems. I think that's related to this, is if you can get something like Dom reconciliation, or rather virtual Dom reconciliation and it's just a built-in piece of the platform, then you can stop thinking about what library you're using for that and whatnot and just continue on.

[0:54:30.3] CM: Okay. Besides a phone, what are the other ways Facebook may use React Native to gain a business advantage?

[0:54:39.2] AP: That's an excellent question for Facebook, I think.

[0:54:43.1] BV: It could be funny to speculate. I can't think of anything off the top of my head. They're already using it at business advantage in the sense that they attract, I think, a lot of developers to their company just based off of their reputation for building innovating open-source software like React Native and React, and Jest and Flow and that sort of thing. That's really, I think, a huge point, because if a lot of people look at — Rather, if you were to look at the current hiring state, or rather even the reputation of companies currently in the Bay Area, I think Facebook is held in very wide regard because of the work that they do in open-source.

I think, even, if you look at what Microsoft has done over the last four or five years, or maybe it's even been less than that, where they've been really making a push to release more open-source software, and that's improved not only the reputation amongst developers, but it kind of trickles down to non-developers, just a general sentiment towards Microsoft, I think. I think it couldn't hurt to keep doing things that demonstrate that you're an innovative forward-thinking company and it kind of creates that perception in their community.

[0:55:53.0] AP: I also think that there's something — It would be irresponsible of me to talk about the business advantage of React Native without addressing the technical implementation advantages that they get out of it. They're able to share code across platforms. They're able to iterate on user interface elements more quickly. It's not like this technology is surely political, they get actual real world day-to-day benefit from it.

[0:56:20.4] BV: This is the issue of being so deep in the weeds that I just assume that's —

[0:56:27.4] CM: This has been a great conversation, Brent and Adam. Thank you so much for joining me on Software Engineering Daily.

[0:56:33.3] AP: Thanks so much for having us.

[0:56:34.0] BV: Yeah. Thank you.

[END OF INTERVIEW]

[0:56:39.7] 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 Symphono.

[END]