EPISODE 627

[INTRODUCTION]

[0:00:00.3] JM: Flutter allows developers to build cross-platform mobile applications. In our previous show about Flutter, Eric Seidel from Google described the goals of Flutter and why he founded the project and how Flutter is built. In today's show, Randall Schwartz talks about Flutter in more detail, including the developer experience of building Flutter floater apps and why he finds Flutter so exciting.

Randall was a fantastic guest because he's the host of a popular podcast himself. It's called FLOSS Weekly. It's a show about open source software, and Randall is also a longtime software developer who is mostly focused on web applications. So it's interesting that he started developing for mobile, basically because he was so excited about Flutter.

Like many developers, Randall has stayed away from mobile development in the past, because if you write a mobile application historically, you've had to build on iOS and android apps separately and that's a large upfront cost. Flutter reduces that cost by allowing users to develop mobile apps for iOS and android with a single code base.

Randall has been podcasting about open source software for 12 years, so he brings significant historical depth to this conversation, and he's also been working with Dart. For several years, Dart is a language developed by Google that is used within Flutter. So there is a whole lot of context that Randall brought to the show and I really enjoyed speaking with him about Flutter.

I want to mention that we are hiring at Software Engineering Daily. We're looking for an engineering journalist, a researcher, a videographer, a writer, and several other roles, and you can find these along with other jobs at softwareengineeringdaily.com/jobs. We would love to see your application.

[SPONSOR MESSAGE]

[0:02:04.6] JM: Cloud computing can get expensive. If you're spending too much money on your cloud infrastructure, check out DoIt International. DoIt International helps startups optimize the cost of their workloads across Google Cloud and AWS so that the startups can spend more time building their new software and less time reducing their cost.

DoIt international helps clients optimize their costs, and if your cloud bill is over $10,000 per month, you can get a free cost optimization assessment by going to D-O-I-T-I-N-T-L.com/sedaily. That's a D-O-I-T-I-N-T-L.com/sedaily. This assessment will show you how you can save money on your cloud, and DoIt International is offering it to our listeners for free. They normally charge $5,000 for this assessment, but DoIt International is offering it for free to listeners of the show with more than $10,000 in monthly spend. If you don't know whether or not you're spending $10,000, if your company is that big, there's a good chance you're spending $10,000. So maybe go ask somebody else in the finance department.

DoIt International is a company that's made up of experts in cloud engineering and optimization. They can help you run your infrastructure more efficiently by helping you use commitments, spot instances, rightsizing and unique purchasing techniques. This to me sounds extremely domain specific. So it makes sense to me from that perspective to hire a team of people who can help you figure out how to implement these techniques.

DoIt International can help you write more efficient code. They can help you build more efficient infrastructure. They also have their own custom software that they've written, which is a complete cost optimization platform for Google cloud, and that's available at reoptimize.io as a free service if you want check out what DoIT International is capable of building.

DoIt International art experts in cloud cost optimization, and if you're spending more than $10,000, you can get a free assessment by going to D-O-I-T-I-N-T-L.com/sedaily and see how much money you can save on your cloud deployment.

[INTERVIEW]

[0:04:28.2] JM: Randall Schwartz, you are the host of FLOSS Weekly. Thanks for coming on Software Engineering Daily.

[0:04:32.8] RS: I'm very happy to be here. Been listening to your show for a while. So, yes.

[0:04:35.9] JM: Oh, I didn't know that. Okay. Well, I've been listening to your show since college. So I think we're mutual fans.

[0:04:40.9] RS: Awesome.

[0:04:41.7] JM: I brought you on to talk about Flutter and Dart, and we did a show last week about Flutter. It hasn't aired quite yet, but after doing that show, I can understand why you are excited about this project. What don't you explain from your perspective, why is Flutter and exciting project?

[0:04:59.4] RS: Okay. Well, let's go back a little bit, because we have to talk about the history of Dart as well, because that tails right into it. So I've been tracking Dart I think for about six years now. Dart was originally created by Google as sort of a potential replacement for Google Web Toolkit, GWT they call it, which is basically using Java on both server-side and client-side. A subset of a Java that could be compiled down to JavaScript so that they could do their typically single page applications, and that's the kind of thing that you would see if you went to Google Mail. How you press a button on Google Mail, you're not taken into an entirely different webpage. It stays on the same page effectively, although different parts of it show up and disappear. That's a single page application, SPA. So that's all the new rage now these days.

So Google created Dart as a way of having a much more modern language and easier to use language in place of Java as the client-side compilation. Java is semantically huge, and so they had to stay with a semantic subset to be able to go to JavaScript. One of the unique features of Dart is that it is semantically, entirely encompassed for JavaScript. There's nothing in Dart that cannot be compiled to JavaScript. So you write a program in Dart, you're happy with that.

Okay. Originally, development for Dart was done with – For a client-side, was done with a special VM inside Chrome, embedded in Chrome, called Dartium, of all things, and that was interesting. But one of the things that the Dartium did pretty early on, and I'm really happy that they did this, is they decided, “Well, we’re ere not going to get the Dart VM in Safari. We’re not going to get a Dirt VM in Firefox. We’re not going Dirt VM, especially in Internet Explorer.”

So they decided instead they would focus their efforts on building the best possible Dart to JavaScript translator that will work incrementally, that would work with any browser, although typically only works with Chrome, but that's okay. I don't care about that. It’s developed in Chrome. That was seen sadly as a deathknell for Dart, because they said, “Oh! Well, Dart is never going to be in every VM, and even abandoning their own browser, Dart is dead,” and that was about three or four years go.

That was sad, because Dart was not that. It was still being used heavily inside Google. Hundreds literally of Dart programmers within Google right now, at least the numbers they tell me. So Dart was alive and well and Google's two biggest applications where they make their money, AdSense and AdWords were converted to be away from GWT, converted over to be completely AngularDart.

[0:07:41.7] JM: Just to go back to what you said about the sort of runtime model for Dart. So, originally, they were thinking, “Okay. Let's make Dart, and then we’ll embed a VM in whatever browser is going to consume Dart and the VM will be able to turn Dart into JavaScript on the fly?”

[0:07:57.4] RS: No. It would actually run –

[0:07:59.0] JM: It would just run Dart explicitly. So Dartium process Dart on the fly and turns it into pixels on the screen, whereas what they ended up having to do was to make an interpreter for JavaScript, from Dart to JavaScript.

[0:08:11.1] RS: Right. So their server-side [inaudible 0:08:13.8] development tool would translate Dart to JavaScript before it downloaded. What the biggest change has been in the last couple of years is that by implementing Dart strong mode across the board, which is Dart shifted them from being a language that was sort of loosey-goosey, like, say, Python Pearl, that sort of thing, to being a language which is strongly typed. Every variable, we know the only things that can go into it.

This simplifies the cross compilers, because then the compiler can know specifically that this is only going to have an array events. So it can do that very, very much for precisely. So by doing that, they were also able to shift from being an all or nothing Dart to JavaScript compiler, where it had to download all of the JavaScript every time to being an incremental compiler.

I'll tell you, I just started playing that last week, and it's so fun because you can change one little part of your class and it just sends down the incremental parts that it needs, and it's running in straight chromium, or strict Chrome. So I don't have to worry then about the development times, development cycles, because I want to get the same performance that I got with the old Dartium set up, but it's going to be now with modern Dart.

Okay, this is all lead in to what's been happening recently. So the Chrome team was challenged with the idea of how can we make Chrome faster? So they started by ripping out almost everything. The thing mostly makes JavaScript and WebKit slow is that the CSS rules over the years have gotten crazier and crazier to the point where multiple rendering passes down and up have to be made to comply with standard JavaScript standard CSS.

So the Chrome team said, "What if we could start over and not implement all of CSS? Not implement all of the layout rules, like Flex and things like that, RenderBox? Things like that." So they ripped all that out and they got a much better experience. Okay. But it still wasn't enough. So they had been paying attention to what the Dart team was doing and said, "What if we use a language like Dart as our implementation language?"

So they went to the Dart guys and said, "Here's what we need. We want to put this in." They took the rendering engine from Chrome, which is called Skia, S-K-I-A, which is basically a way to control every pixel on the screen. So that's what Chrome uses to be able to draw on the screen, and there's also a text engine, whose name I can't recall right now. They took those two parts of Chrome and they said, "Let's start with that and let's build Flutter by building Dart all the way up from there."

So what Flutter essentially is, is using the Dart language to control every pixel on the screen using Chrome's native rendering engine and Chrome's native text engine. That's how it came around. But they worked with the Chrome, worked with the Dart team to be able to build it in order to deploy stuff in the iOS store at least, not necessarily android store, but to deploy stuff in the iOS store, it has to be not a VM. It has to be not JIT or whatever.

Dart's primary design prior to that was a JIT. So one of the things that they did – Just-in-time compiler. One of the primary things they did was they got the Dart team to build Dart as having an AOT, ahead of time compiler. So when you're dealing with stuff in Flutter, you're actually dealing with stuff that is completely compiled to ARM code. So this makes it even faster than Java apps deployed on android, because that has to go through the Java VM to be able to render things. So Dart is actually extremely performance. 60 frames a second is typical, and this is part of also why I'm excited about this.

[0:11:55.0] JM: So go over that one more time. When Dart runs on the web versus when it runs in the context of Flutter, what's the difference?

[0:12:04.8] RS: Okay. So Dart on the web is translated through the modern compiler, which has multiple modes now, the DDC they call it. That translates to JavaScript. So you're running it entirely in JavaScript. You're talking JavaScript objects, JavaScript renderings, JavaScript the DOM model.

[0:12:22.6] JM: And that compilation is done before your code ships to the user.

[0:12:26.0] RS: Yes, right. Recently, again, it's incremental, which means that if I change – In a large application, I change a couple of methods, I'm only sending that piece of it back down to the browser. So that's the browser mode. That's' truly totally available, totally being used by Google. Something I'm sort of ignoring for now, because I want to be on the Flutter side for a while.

So on iOS and android, Dart operates in two different modes. One is the JIT compiler. So, basically, for development, an entire VM is downloaded to the browser, or to the device, or to the simulator, and incremental parts of it are sent over and compiled just in time. So you can't ship that though, because at least the iOS store and probably the android store both say, "We don't want JITs. We don't want to be able to have side-loading while we're live."

So when you go to deployment, you want to ship it up to the stores. There's another button you press and it becomes AOT. This means that the entire thing is gone through. We do tree shaking to make sure we're only including the bits that are actually used by this application. So that means that binaries are smaller, and then that is completely compiled into ARM code, native ARM code for both iOS and android.

So unlike on android where you're typically running in the JVM, you're actually running native ARM code on the android. So it's actually better even if – If all you're going to ever do is android development, Flutter is still the better choice.

[0:13:59.6] JM: Incredible.

[0:13:59.5] RS: Just because it's completely ARM code at this point, like native apps, very native apps. And on iOS, same thing. Compiled all the way down to ARM code and then talking to the native platform API.

[0:14:12.3] JM: So you said in development mode you're going to be using a JIT with Flutter. But when you actually ship it to the app store, you're going to ahead of time compile everything into the native ARM code.

[0:14:25.1] RS: Yes.

[0:14:25.8] JM: So is that to say that the just-in-time compiler was written exclusively for development mode?

[0:14:32.5] RS: It already existed. It was basically how the server-side code runs. So if you want to run Dart on server-side, which is also from command line apps you would run it. That essentially is also a JIT, a JIT VM. So that's already there. That's been in place for years.

[0:14:50.2] JM: Okay, got it. Now, to zoom out a little bit, Flutter is for writing cross-platform mobile applications and we've had past cross-platform UI solutions. A lot of these have been built using some sort of JavaScript bridge. I went over this a little bit with the Flutter show that I did last week, but could you contrast Flutter with the previous cross-platform solutions and just give little emphasis for why this is such an impressive and important project?

[0:15:21.9] RS: Okay. So, originally, the most classic of writing for both iOS and android has been to write a JavaScript application that would talk to the native WebKit. So the WebKit in iOS and android are fairly similar. Basically, you're manipulating the DOM with your JavaScript. There has to be some sort of way to gap across to getting input events.

So, for example, accelerometer, or GPS, or anything like that. So there are ways where those events are sent into your JavaScript code, injected in your JavaScript code. But basically what you're doing with the JavaScript is manipulating the DOM.

Okay. And so the problem with that, again, is the same thing the chrome team come up with a couple years ago before we started Flutter, is that that's an expensive thing to do. The DOM has become so rich, so powerful that the semantic bundling is dangerous. It's just too expensive to use consistently, and that's why we get slower and slower web application, for example.

So the first step to fix that was – I forget what projects this was. But basically instead of using WebKit to do the rendering, there was a bridge inserted between the JavaScript code and native widgets. So in other words, on iOS, you would use the widgets that look like the natural iOS app widgets. On android, you would use the android native app widgets later in life. Of course, those all became material widget.

So you would then have to design your app so that it would be compatible with both to some degree that was provided by the platforms and made it better. Okay. The problem with that bridge is that you're still going from the native input events through a JavaScript bridge then back through JavaScript to control the native widgets.

So you go through two different levels of bridges and the responsiveness was appropriately diminished, because you're going – Anytime you change semantics, anytime you change semantic layers, you have the problem of impedance mismatch. That's the word I'm looking for, impedance mismatch.

Okay. How Flutter differs from all that is that the input events are being delivered through local Java, or Kotlin, or Swift code directly into Dart. It's dart all the way up and all the way back down to the point where Dart is then talking and drawing pixel for pixel, everything at once using Skia. So there's no double bridge problem. There're zero bridges. It's basically local events, immediately injected into Dart, immediately written back as widgets in the display screen.

[0:18:06.8] JM: What does this do for the developer experience of creating native cross-platform apps?

[0:18:13.7] RS: Well, for one, since you're controlling with the same construct, the same dark code, you're able to get pixel for pixel control on both iOS and android and its complete control. You want to have a button that blinks green when you press on it, and there's nothing like that that does it right now. You can basically drill down. The code is entirely Dart. The IDE's, you can select a class of something you're trying to implement and it goes down shows you the exact source code, because it's all dart all the way down to the pixel level.

So you can see how the current widgets are implemented. You can subclass those. You can override those. You can even modify those if you don't like the way they work. So while the default widgets sort of use a material interaction, you can basically do anything you want. I mean, It's completely controllable all the way down to the pixel level.

[0:19:09.2] JM: I haven't done very much mobile app development, but my intuition is that there is tooling in each ecosystem. There's tooling in the android ecosystem. There's tooling in the iOS ecosystem. There's probably some components that Apple has developed some components that Google has developed for android that make it easier for developers to have kind of a drag-and-drop experience for building their UIs. Does Dart have to re-create all of these stuff from scratch for their development experience?

[0:19:39.7] RS: Actually, it's interesting. It's interesting you asked that. So the thing about it is that the Flutter team has been very, very interested in completely mimicking the native android experience, including – And this is amazing. They took high-speed video camera images of people doing things on an android phone and they reversed calculated what the formulas would have to be to get that exact acceleration, that exact view, everything.

What's funny is a sent that back to the android team, they went, "Oh! That's an even simpler formula than we were using." So it's been an amazing experience to see the Flutter team working together with the android team to make sure that's there. So, yes, they're very, very interested on the android side especially, because that's their bread-and-butter right now. They're very interested in making sure that everything you're used to. But here's the plus side of this. Flutter ships with material components according to the material design. In fact, the Flutter team is a first-class customer of the material team. So they work very closely together.

Okay, so you can ship material helps on platforms for android that go way before material was ever introduced on android, which is really amazing. It means you can have the same experience all the way back. Whereas, typically, you had to have some sort of shims and stuff as you went older and older, releases. Same thing with iOS, you had to have shims to go back and back, but the only thing that's really running with Flutter is the rendering engine and the text engine and that's going to be the same no matter what platform you're on.

[SPONSOR MESSAGE]

[0:21:40.2] JM: At Software Engineering Daily, we have a web, we have an iOS app, an android app and a backend that serves all of these frontends. Our code has a lot of surface area and we need visibility into problems that occur across all of these different surfaces. When a user's mobile app crashes while playing a podcast or reading an article, Airbrake alerts us in real time and gives us the diagnostics that let us identify and fix the problem in minutes instead of hours.

Check out airbrake.io/sedaily to start monitoring your apps free for 30 days. Set up takes only a few minutes. There's no complicated configuration needed, and Airbrake integrates with all of your communication tools from Slack, to GitHub, to JIRA, and it enhances your current workflow rather than disrupting it.

You can try out Airbrake today airbrake.io/sedaily. If you want to monitor and get visibility into the problems that may be occurring across your application, check out Airbrake at airbrake.io/sedaily. Thank you to Airbrake.

[INTERVIEW CONTINUED]

[0:23:00.2] JM: Are there some limitations to the Flutter developer today? Is there some subset of the totality of mobile applications that you could build that are exclusively that's the subset that maybe you should write apps for Flutter and maybe the more expressive, the more ambitious apps you should stick to native code?

[0:23:20.5] RS: Well, Flutter is primarily intended at this moment for being beautiful 2D interaction. So it's not a 3D thing yet. It's not going to replace unity, because it's not designed to do that yet. Not say it's not going to come along, but it's really about building the perfect interface for your typical text and visual based opportunities.

Now, just recently somebody released a sort of a 2D animation thing that was really kind of cool. So that got some play at Google IO. I haven't had time to watch the detailed videos on it, it but looks like that's going to be interesting where you can start doing animations and stuff like that with – But, again, still 2D. Not 3D yet.

[0:24:03.4] JM: Have you talked to people who had invested a lot in using React Native in their apps and have you heard any responses from them on what they think of Flutter?

[0:24:14.2] RS: I've read many, many blog posts about that very thing. And, again, React Native is great, except, first you're writing in JavaScript, which is a legacy language, has its own quirks. I won't ask you to name all the different ways you can say true and false in JavaScript, because that would probably embarrass both of us.

In Dart, there is exactly one way to say true, true. Exactly one way to say false, false. If three, it blows up. That is not true or false value. So there’s that. So you have to write in JavaScript. Not only that, that still has to cross this bridge. So React Native, even though you're compiling to ARM code, you're still going to have to cross this bridge that talks to native widgets. I forget whether React Native is Native Widgets or WebKit. I think – Yes, Native Widgets. But it still has to talk to that and you still have to deal with the iOS versus android problem. So there's that too.

So the blog posts I've seen of the people who have shifted from React Native over to Flutter have been very positive. Of course, I do know. Maybe I’m not reading the ones that are negative, but I do say probably pretty positive.

[0:25:23.7] JM: You were using Dart before you started using Flutter, and you've been programming for, I think, like 30 or 40 years, something. You've been programming for really long time. So you have some history on programming language preferences and you're capable of contrasting languages between each other. How does Dart compare to the other languages that you've used?

[0:25:46.8] RS: It sort of makes me feel like I did in my Smalltalk days. So I was programming in Smalltalk in 1981. So way back to the first release, which is 1980, I was actually working for a company that had access to it in ’81. Dart feels a little like that, and that everything is an object, but it's got familiar syntax. I mean, it’s similar. Anybody who knows Java or C# is going to be able to stare at Dart and go, “I know this. This is mine. I know this,” and that's deliberate.

Google was not set out to be innovative with this language. I mean, they already had to Go, which was covering a space for them that they really liked, but Dart was about something simple and something with a semantic subset that will always be translated to JavaScript. So that keeps the sort of the corners out of the way.

But out-of-the-box, it has promises and streams and strong typing, the kind of typing where you can specify that this takes a subset of that, “Oh, generics.” So it has that out-of-the-box, and being able to build a large application in the small and share stuff, and also out-of-the-box, we’ve got the infrastructure, the pub, pub.dartlang.org is equivalent in Perl of the CPAN and already has thousands of reusable modules, some of which are reusable even for Flutter and for server-side and for client-side. It's already got that ecosystem in place, and there are quite a few Dart programmers. Dart hasn't hit 2.0 yet. Flutter hasn't 1.0 yet. But my sources reveal that the basically those will happen pretty much the same week. It's coming soon.

[0:27:32.3] JM: Dart also has a garbage collector. That's pretty important to Flutter. I think JavaScript is a garbage collected language, right?

[0:27:39.9] RS: Yes.

[0:27:40.5] JM: How does the Dart garbage collector compared to the JavaScript garbage collector?

[0:27:44.6] RS: Well, in Dart to JS, for client-side, it is the JavaScript collector.

[0:27:51.1] JM: Oh yeah, okay. So you're just using JavaScript. So this is only important for server-side development.

[0:27:55.1] RS: Yeah. So server-side or Flutter side, one of the things they did was they optimized the Flutter garbage collector so that it would be more responsive for interaction. So it basically works more on the idle time. It runs, I think, in a separate thread and it just works really well. So yeah, it's optimized for the best possible behavior on mobile apps.

[0:28:18.4] JM: So when you're compiling to ARM code, they must have a – I guess, the garbage collector can run in that case, because it's Dart to ARM. You're saying since it's not going through the JVM, the language is more performant. Does that include the garbage collector?

[0:28:35.5] RS: Yes. So essentially there's some small part of the VM that is also shipped, and part of that of course is the management of promises and streams and threads and garbage collection. So that's all – That part of the VM is a constant that goes into the native ARM code.

[0:28:53.3] JM: Okay. Interesting. So Dart, it also has a lot of nice asynchronous programming features. I know my front-end JavaScript code frequently involves callback health or asynchronous network programming. Does Dart solve any of the callback hell that you can have in JavaScript?

[0:29:13.7] RS: Well, it's as I said before, it basically promises and streams are first class elements of language. So if you're familiar with – I think they call it futures in JavaScript. So what this does is it enables the ability to say, "I can return a value from my subroutine that is essentially a promise to say, "I will eventually return the real value."

The caller will know that, because it's strongly typed, that he needs to eventually call either await, A-W-A-I-T, await, or call it attacking on a .then, which is basically a closure that will be executed once the promise is fulfilled. So yes, that avoids callback hell, because we're basically passing around the future.

[0:29:59.4] JM: What apps have you developed in Flutter?

[0:30:02.6] RS: Oh, I wish I could say there were many. I am working on probably about three or four things right now, but there's nothing of mine in the play store yet. So, sorry.

[0:30:11.7] JM: Had you done mobile app development before, like in iOS or android?

[0:30:15.8] RS: No, and this is really the big deal for me with Flutter, is that I am empowered now to develop on mobile. I mean, I've got three apps on my phone right now that I wrote, that didn't come in the store. They're on my phone right now that I wrote. I had looked many years ago at developing for iOS and realized I have to learn the whole infrastructure and have to learn Objective-C, which wasn't bad.

I mean, again, my Smalltalk background made Objective-C sort of enticing, but not exciting. But if I ever wanted to go to android and cover the entire 80% of the mobile market, I thought, "Well, let me have to learn Java," and I don't really like Java. I've never liked Java. I thought, "Well, if I'm going to really develop from all the markets –" And not only that, not only Java. You'd have to also learn the whole infrastructure on that side. So different tooling, different application interface, things like that.

I am empowered by Flutter to be able to write code that does exactly what I want and runs on both iOS and android without doing more than pushing a couple of buttons. So I am empowered. I'm definitely –I've got like probably three or four ideas immediately that I want in the store tomorrow, but my timing is, of course, tough.

[0:31:35.8] JM: That's awesome. So what about the website of things? I guess it doesn't – You don't have that same psychological barrier to thinking about, "Okay. Now I'm going to be developing a Flutter app for the mobile platforms and then I'm going to have to figure out some other web front end if I want also port this to the web."

[0:31:54.2] RS: Well, so here's the thing also, is that there are some very good server-side frameworks. I'm thinking of Angel. So Angel is really good. It allows you to build the same tooling to be able to have your mobile app talk to a RESTful interface and also have your AngularDart app talk to RESTful interface and to the degree that you have business logic that does not need to talk to either a WebKit or do direct I/O. In other words, it only does RESTful or whatever like that. You can share that logic across mobile and web applications, which is really, really nice.

[0:32:39.1] JM: What do you anticipate the next move by Google being or when Google decides to make it even easier to have Flutter, your Flutter apps, run in the web? Do you anticipate – is there any clear line that you can draw on your head, like what is that going to look like when you can just run a Flutter app on the web easily?

[0:33:00.0] RS: There's no motivation internally that I'm aware of, because there's no business case where Flutter will work in a browser. That's not the case, because Flutter is basically talking strictly to the Chrome Skia. Now, that's not to say that third parties haven't done this. So they've actually built electron apps that are basically running Flutter, because all Flutter needs is a canvas.

So once you have the canvas, you can paint on it. So there are third parties doing that, but that's not in Google's business model right now. It's like – So where Google is going with this, is they're clearly committed to Flutter. Hundreds of developers working on Flutter internally, I know this. So hundreds of developers working on Flutter. They're committed then to the iOS and android platforms. They're committed to AngularDart, because they replaced AdSense and AdWords away from GWT over to Angular.

So that's absolutely in their bailiwick, but they're not really interested in having desktop apps or having web apps speak Flutter, because they already have the AngularDart solution. So that's probably how they're going to go, and that AngularDart solution of course talks to the DOM. It's basically Angular, but written in Dart.

However, I will say, just to throw something else in here, that this thing called Fuchsia, which isn't well-defined yet, but seems to be the possibility of an ultimate android replacement, and why that's important to Google is that Google and Oracle are at odds about the Java situation. Because Oracle owns Java at this point, owns the Java SDK. Google had made a handshake deal with the head of Sun, before Sun got bought by Oracle to say, “Yeah, we can use Java, right?” “Yeah, okay.” So that all went down, right? But they never got it in paper.

Now Oracle says, “Ooh! We own these interfaces,” and they've gone to court about it two or three times, and Google does not want to be in the position of having to pay royalties forever for every copy of android shipped, which they are potentially at risk for right now.

So what Fuchsia seems to be is a new operating system from the ground up not using the Linux kernel, not using any job interfaces, but primarily designed for small devices, phones, IoT, your display in front of your refrigerator, that kind of thing, right? And it's done primarily in Flutter right now. So Flutter might be the future of Google's small platform interfaces.

[0:35:58.2] JM: Have you talked to anybody with direct knowledge of Fuchsia, or are you just talking to people who have dissected what code is out there in the public?

[0:36:06.0] RS: I can't exactly answer that question.

[0:36:09.2] JM: Okay.

[0:36:09.5] RS: So let me just say that.

[0:36:11.6] JM: Got it. Well, can I ask, have you yourself looked at the code in detail, the Fuchsia code?

[0:36:16.9] RS: No, I haven't. I run OS X on my laptop. I don't run Linux, and I need the Linux box to be able to start playing with that stuff.

[0:36:24.3] JM: Right. Yeah, I have seen some YouTube videos of people playing around with it, and it looks like people trying to investigate, like after an alien has landed and they're just like looking around and poking around and don't really know what's going on.

[0:36:36.7] RS: One of the problems, of course, is that Google, being a publicly traded corporation, cannot deliver future looking statements without getting into trouble. So whatever they're doing – That also frustrates me about the schedules for, say, Dart and Flutter, is like they can't say, "Yes. We'll be finished – We've going to have our 1.0 release in June." They can't say that, because it's a future looking statement each for the statement. However, I know people. So, yeah.

[0:37:06.5] JM: Right. You're pretty sure that this is about the legal battle. It's not about some improvement that needs to be made to – Or improvements that could be made to the Linux kernel that would cause Google to just say, "We got to throw out the Linux kernel and completely rebuild something from scratch."

[0:37:24.6] RS: No, I can't say that for sure, but you look at kind of the minimal android footprint, and it's still not the thing you want running on the device that's on your refrigerator, or Google Home. Okay. So you've got this Google home device. Why does it need to run all Linux? That's silly.

[0:37:42.3] JM: So it's silly because of the JVM, or – I mean, I don't quite understand.

[0:37:47.2] RS: Well, because there's a lot of things that happen in Linux that are really relevant to just a box that's going to sit there and listen to your voice and send it upstairs and then come back with a response. IoT – It's too much of a footprint for IoT.

[0:37:59.6] JM: I see. Have you ever looked at Micro kernels?

[0:38:03.5] RS: Well, that's essentially what Fuchsia is. They're starting with a brand-new kernel and it's supporting – Now, I have to say, part of it that's interesting in the last few weeks, what I've read, is that Fuchsia may be supporting android binaries. Now, if Fuchsia's running on

Chrome books and also running on the device from a refrigerator, but it can support android binaries, this is going to be interesting.

[0:38:31.5] JM: I’m sorry. I said microkernels. I meant unikernels, which is something different. I don’t know if you looked at those.

[0:38:36.6] RS: Not so much.

[0:38:37.8] JM: Not so much. Yeah. Okay. What do you think is the future of the consumer operating system based on what you have seen Google putting out and based on your understanding of the landscape?

[0:38:46.7] RS: Well, again, I'm looking at – We’re going to keep moving to different devices. I mean, iPads are already proving that people are very happy not having like a full-sized laptop when they want to get their work – And I think we’re going to see the need more and more to have that kind of device. I think Fuchsia is probably a good piece in that. I think being able to program mobile apps to get smarter and smarter with something like Flutter is also a good way to keep doing that. I mean, it's upgraded from the iPhone's 6s to the iPhone 8+ recently, and already I'm using my phone for more and more things.

I think this is an interesting trend. I can't predict the future very well, or also be rich, but I do see what Google's trying to do here is enter the space that is going to enable them to be a key player in this, and I really like that. That's why I'm all in on this as I've told people multiple times at Google, is that I’m all in on this. This makes sense to me. I am as excited about this as I was about Perl in 1990, and that says a lot, because Perl went huge after 1990. It feels the same for me right now.

[0:40:01.8] JM: And at a fundamental level, is that because for the longest time, one of the axioms of software development is that you have to develop custom software for every user interface surface. If you're going to develop a Windows app, you got to develop a special UI there. iOS mobile app, you got to develop special UI there. But this is actually – The promise here is really to unify all of that.

[0:40:28.6] RS: Well, it's not just unification, because that's been done. React Native did that. What I like about this is that we have ARM code, we have native code all the way down controlling every pixel. So it's high performance. Even if you never do anything except for one or the other platform. This is still going to be a better solution. It's a better language to develop in. The IDE’s a great. You're compelling to ARM code. You’ve got control of every pixel. It's entirely a reactive, not to confuse with React Native, but a reactive widget tree all the way down designed by people who – This is their third or fourth system. This is not a first system application. This is people who have had to do this over and over and over again and said, “Boy! If we could only do X,” and they did X, and they did Y, and they did Z, and this is Flutter.

[0:41:28.2] JM: Explain that reactive widget tree. What do you mean by that?

[0:41:30.3] RS: What I mean is that – So the design internally is that everything is a widget, even layout is a widget. So there is a center widget that takes it’s one child and puts it right inside of center. Okay. So everything that’s widget all the way down to the point where you’re rendering like some text or a picture or something like that.

Now, in the widget tree, which is basically just a hierarchical tree, there are stateful widgets and there are stateless widgets. The stateless widgets in theory can be completely destroyed and re-created without losing anything, and they might have that done if something in their parent changes the size of where they should fit or something in one of their children change the size of where they should fit.

So it's a one pass all the way down the tree, rendering all the widgets. Stateless widgets can be thrown away and practice are actually recycled, but that's transparent to you. But in terms of thinking about it, you’ll always have to believe that a stateless widget might be completely thrown away.

On the other hand, stateful widgets have an associated state object with them, and when you want to change something about that state, for example, a number should be 17 instead of 16. You call a set state on that stateful state and it informs its widget, “We might have to re-layout,” and then that triggers the whole state rendering tree.

So everything is done by updating state that are associated with stateful widgets that are in part of a tree with both stateful and stateless widgets. It's a very simple model and a very beautiful model and it works really well.

[SPONSOR MESSAGE]

[0:43:21.4] JM: Nobody becomes a developer to solve bugs. We like to develop software because we like to be creative. We like to build new things, but debugging is an unavoidable part of most developers' lives. So you might as well do it as best as you can. You might as well debug as efficiently as you can. Now you can drastically cut the time that it takes you to debug.

Rookout rapid production debugging allows developers to track down issues in production without any additional coding. Any redeployment, you don't have to restart your app. Classic debuggers can be difficult to set up, and with the debugger, you often aren't testing the code in a production environment. You're testing it on your own machine or in a staging server.

Rookout lets you debug issues as they are occurring in production. Rookout is modern debugging. You can insert Rookout non-breaking breakpoints to immediately collect any piece of data from your live code and pipeline it anywhere. Even if you never thought about it before or you didn't create instrumentation to collect it, you can insert these nonbreaking breakpoints on the fly.

Go to rookout.com/sedaily to start a free trial and see how Rookout works. See how much debugging time you can save with this futuristic debugging tool. Rookout integrates with modern tools like Slack, Datadog, Sentry and New Relic.

Try the debugger of the future, try Rookout at @rookout.com/sedaily. That's R-O-O-K-O-U-T.com/sedaily. Thanks to Rookout for being a new sponsor of Software Engineering Daily.

[INTERVIEW CONTINUED]

[0:45:26.1] JM: I've heard the Flutter people emphasize composition over inheritance. Can you explain what composition means in the context of Flutter development?

[0:45:36.2] RS: It means that instead of, say, subclassing from a text box, I would include a text box as part of my children and then control its ideas.

[0:45:49.3] JM: Okay, got it. How do you anticipate web assembly changing client-side development? Does that relate to this conversation at all?

[0:45:58.6] RS: Sure. That's been discussed. I've seen discussions on it. The issue with web assembly is that, as I recall, Dart to JS may eventually target web assembly, but there are some elements of web assembly that still did not encompass the things that Dart to JS uses of JavaScript. So they're waiting for the next or next, next rev of web assembly before they even start work on that. Again, not their primary target. As long as Dart to JS does the job for all modern browsers, web assembly is not going to give them much more.

[0:46:36.7] JM: I'm not sure if I completely understood that.

[0:46:39.0] RS: So the thing is – Right. So web assembly gives you access to essentially a machine language, as I understand it, machine language sort of version of that. There are still elements of the semantics of the browser that aren't exposed through that, the Dart to JS users.

[0:46:56.7] JM: Oh, okay. I see.

[0:46:58.1] RS: So it's still going to be better for them to go, "Well, here are some JavaScript."

[0:47:02.5] JM: I see. So you're saying web assembly is a lower level language that, today, different languages can cross compile too. So you can compile your C++ code to your Rust code to web assembly and it will run in the browser, which is amazing, like running your Rust or your C++ code in the browser. That's fantastic. If you want it to run Dart to J – If you want it to run dart code on web assembly, if you want to compile Dart to web assembly, you would first have to compile it to JavaScript, I guess, today?

[0:47:37.6] RS: No. No. No. No. But the thing is there's – Yes. You could do it that way too, but the big thing is what people are asking for, of course, is direct Dart to web assembly, but the problem with that thing is it's not – First off, from what I understand, there are some semantic still missing from web assembly that are needed to do what Dart needs of the target platform. But also because that's still missing, there's no motivation for anybody at Google to do that.

Now, it might mean third-party can do that, but you got to understand that Google has – Google's got to pay the bills and they're only going to do the things with Dart and Flutter and Fuchsia that make sense to them in terms of an overall business strategy, and that's not one of the things that they're aiming for.

[0:48:28.6] JM: And that's because –

[0:48:30.4] RS: Because Dart to JS is already good enough.

[0:48:32.2] JM: Oh, okay. Dart to JS is already good enough, and that is what Angular is using.

[0:48:37.5] RS: Yup, AngularDart. Yes.

[0:48:39.2] JM: AngularDart.

[0:48:40.2] RS: Which is not Angular. That's two separate projects now. They were originally working from one typescript base, but they realized that there were things that the Angular JS team needed that were not needed when you were in Dart because the language is better. So they actually split. So there's no intent at this point of synchronizing and keeping in sync Angular JS and AngularDart. Some call that a loss, but I think AngularDart in the meanwhile has moved in some really great corrections. So I'm happy that I get to play with AngularDart.

[0:49:16.9] JM: Who is AngularDart for and who is Angular JS for?

[0:49:20.6] RS: Well, Angular JS is for people that want to code in JS, JavaScript, and love the classic. Although here's the thing, Angular JS has gone through a couple of revs that have actually upset a bunch of people, because they went through some incompatible revs, right? AngularDart has done similar things, but AngularDart is what Google is building, AdSense and

AdWords and a bunch of other properties in. So AngularDart will be around tomorrow. We can guarantee that.

[0:49:50.8] JM: And they made that shift because the whole idea is Dart is preferable to JavaScript.

[0:49:57.2] RS: Well, for the people who are building that stuff. So AngularDart really came out of the people who wanted to stop using GWI and I don't blame them. That's a crazy thing. Coding in Java for both server-side and client-side. I hate Java in the first place, as I said before.

[0:50:13.0] JM: Sorry. What is GWT?

[0:50:14.5] RS: Google Web Toolkit. It's been around for about 15 years. One of the earliest sort of things that allowed that to happen.

So AngularDart is the heir apparent for GTW. So I don't know that necessarily that Google is updating every app from GWT to AngularDart, but that would be a nice goal. That would be a straightforward goal. Yeah.

[0:50:37.7] JM: So the future frontend or user interface development from Google's point of view is Flutter for your various devices, and Angular JS or AngularDart for your web applications.

[0:50:55.7] RS: I don't know that Google is using a lot of Angular JS at this point. I think they're moving everything over to AngularDart that they can, then server-side, Dart on the VM. But this is not also to say that Google is replacing all of their Go applications with Dart. It only – In fact, very few of their server-side things, except maybe the pub, like CPAN equivalent thing and some of their other servers. That's all done in Dart server-side, but mostly Go still has a huge influence internally. This is not a replacement for Go either.

But if you learn to program in Dart, you've got server-side. You've got client-side. You've got iOS. You've got android, you've got Fuchsia. So you got five platforms you can code for.

[0:51:42.6] JM: So Dart has become one of those blessed languages within Google. So I guess now it's what Dart, Python, Java – Is it Go? Those four languages? Those are languages you can write backend applications in at Google?

[0:51:55.9] RS: Yes, I believe so. I'm not a Google person, so I don't know.

[0:51:59.3] JM: Sure. Sure. What would be a good application to write in Dart server-side? Why would you write a web server in Dart versus Go, or Java, or Python?

[0:52:09.9] RS: Well, again, so the Angular framework is really nice. Written by like an 18-year-old kid who's just like been doing amazing work for this stuff, Toby. He's awesome. So he wrote a framework that allows for RESTful API calls and talks directly via ORMs to PostgreS at least. I forget the other one he talks too, SQLite or something like that. But what's great is then if there's business logic that doesn't need to talk directly to the database or doesn't need to talk directly to WebKit, you can take that exact same business logic and put it in your AngularDart application.

So in other words, you can pre-decide whether this set of fields makes any sense and then — before you even send it. So you can do form validation client-side, browser side, and use that exact same logic server-side so somebody can't trick you and say, "Well, I really want this shopping cart to cost zero," right?

[0:53:06.8] JM: Right.

[0:53:07.4] RS: Which is everybody's dream. That's the great thing, is that you can write server-side applications and drop them into any of the cloud providers, and that's all written in Dart. So you're writing in the same language and using the same reusable business classes that you would then use on your AngularDart client-side.

[0:53:28.4] JM: So earlier in the conversation we were talking about the compilation, the state of compilation for Dart on the web and the fact that you can't do on the fly compilation of Dart in a browser, like Internet Explorer, or Firefox, because why would they support that today? Is there reason for Google to sort of lobby for that in the future, to sort of lobby and say, “Let's make Dart more widely supported,” or is that even unfathomable?

[0:54:00.8] RS: I don't think they'll ever need to do that. Part of the effort, starting a few years ago, when they said, “We now recognize that there will never be a Dart VM in every browser,” is that they made the Darts JS compiler be incremental and be better. They got faster. I mean, there're benchmarks from back then till now that show that they were getting better and better Dart code.

In fact, it's funny. One of the benchmarks I believe showed that for this one benchmark they ran, Dart was able to generate better JavaScript than could be hand generated, and it’s because the Dart analyzer could look out and see this was only ever an int and it figured out optimal things to do with that. So Dart to JS, the incremental one, the DDC, the new one. I don’t know what it’s called. I think it’s just called DDC, but the new one is amazing. It allows me to develop entirely in Dart and only load to my browser what I need.

[0:55:06.2] JM: Randall, you’ve given me a lot to think about in terms of Dart and Flutter and I really need to study this stuff a little bit more. I did want to ask you little bit about podcasting, because you're the host of FLOSS weekly.

[0:55:17.5] RS: Yes.

[0:55:18.2] JM: How has podcasting impacted your life?

[0:55:21.6] RS: It's been amazing. I took it on sort of on a whim 11 years ago, I think. I worked closely with my friend, Neil Bauman, who produces conferences on cruise ships, and I started doing geek cruises news is for him, because I was inspired by my friend, Leo Laporte on one of the cruises to start doing this. That was basically me interviewing the upcoming speakers, or recording sessions that happen on the cruises and then doing that.

Then one day, Leo invited me to be a guest on his show taped at a Vancouver, B.C. and I did some little wacky things. I think the first one I did was using the accelerometer of the Mac laptop to play games, and I had been a guest on FLOSS Weekly a few months earlier, and I wondered why FLOSS Weekly stopped after, I think it was the first 13 episodes or so.

He said, “Well, Chris DiBona, he's been bringing me all the guests and he had a baby and she's not around anymore.” I went, “Well, if I brought you a guest, could we start the show back up?” He said, “Sure!” So that's how it got started. 17 shows. Sorry, 17 shows.

So I became the cohost by bringing my friends in and that was a lot of fun. So I remember Josh Berkus was my first guest and I had sent him a preliminary set of questions ahead of the show. One of the things I asked was, “Is there anything unusual I should ask you about that your history that you wouldn't know, that I wouldn’t know?” He said, “Well, I was once a model for a – I was a naked model for an art class.”

So, of course, I asked him about that in the show and it was a lot of fun. I went, “This is kind of cool, that if I asked them, “What should I ask you?” they would know.” So I've had that as my standard operandi for the 10 years I've done it, which is always ask, “What do you want me to ask you?” and that's been beautiful.

Anyway, so more direct to your question. I get to talk to some of the coolest people in the world, just like you do, and it's fun. It's like I get to see what's on bleeding edge. I get to make friends with people all over the world. So you’re probably at times on geek like I am now, because coordinating this stuff is just crazy. Let's please – Please, let's get rid of DST. Please. Oh my God! That causes so much trouble. Especially now that, say, the US and the UK changed two weeks apart. That's just wrong. That's just wrong.

Anyway. But I get to talk to all these great people, but more importantly, and I'm sure you’ve had the same experience. PR people are emailing me now saying, “I need to have my client on your show,” and that's when I knew I became legit press. The moment I realized I was legit press, which is about six or seven years ago, I started applying to go to conferences as press, and you know you get all the same – It's free to you. It's free, you get access to everything. You could talk to everybody. It's so cool being able to go to any conference I want that’s vaguely about open source and not say anything except, “I’m press,” and they go, “Oh, yeah. We know. You’re FLOSS Weekly. You’re fine.”

[0:58:47.1] JM: Yeah. So you ever find that the habit of podcasting changes your interpersonal interactions outside of the context of a podcast, like in your just day-to-day conversations?

[0:59:03.1] RS: In a sense, yes, because for many, many years, of course, because the publication of the [inaudible 0:59:08.8] all those great books that I wrote many years ago, that was how people came up to me at OSCON and other open source conferences and said, “You’re the dude with the books,” but almost always recently it's been, “You're the dude with that podcast.”

So that does change the interaction a little because I had always wondered what next after Parl. Perl was a godsend at the time. I was in the right place at the right time. I won't admit to having any special skills. I just was in the right place at the right time. To have chatted with Leo Laporte 10 years ago to get myself firmly into the podcast realm and allowing me to join his wonderful network, that was great. I see Flutter as being the next game changer for me, that three years from now people come up to me and go, “You’re the one that did all those Flutter screen casts and you wrote that book on Flutter.” That's going to be my next forte.

[1:00:08.2] JM: Well, you’re staking it out. Randall, thanks for coming on Software Engineering Daily. It’s been really great talking to you.

[1:00:12.9] RS: I look forward about to show you just recorded and see how many times they contradicted me.

[1:00:19.0] JM: Yeah, I don’t think they contradicted you at all. I think it will be a great companion, because people in Google can say a certain set of things and external press people could say a larger set of things perhaps, but a lesson form set of things.

[1:00:32.6] RS: Awesome. Awesome. It’s been an honor and privilege. I’ve listened to show a lot. I can't say since the beginning, because what do you got? Like 700 shows now? Something like that? So, anyway, it's been an honor. Thank you for inviting me on your show.

[1:00:45.3] JM: Thank you.

[END OF INTERVIEW]

[1:00:48.1] JM: Raygun provides full stack error, crash and performance monitoring for tech teams. Whether you're a software engineer looking to diagnose and resolve issues with greater speed and accuracy, or you’re a product manager drowning in bug reports, or you're just concerned, you're losing customers to poor quality online experiences, Raygun can provide you with the answers.

Get full stack error and performance monitoring in one place. The next time you're struggling to replicate errors and performance issues in your code base, think of Raygun. Head over to softwareengineeringdaily.com/raygun. Get up and running within minutes and dramatically improve the online experiences of your users.

Thank you to Raygun for being a sponsor of Software Engineering Daily, and if you want to support the show while also checking out Raygun, go to softwareengineeringdaily.com/raygun.

[END]