EPISODE 1340

[INTRODUCTION]

[00:00:00] KP: Getting a computer program to run the same in different environments has been a recurring problem since the earliest days of software systems. Software version itself, the versions of your dependencies, the hardware configurations, and even the CPU instruction set differences are just a few examples of challenges engineers have faced to get their software to run in different settings.

A core promise of the Java programming language has always been its ability to run anywhere. If your system can run the Java virtual machine, the JVM, then it can run Java bytecode in a way that will be invariant to lots of other factors. Now, Java was created long before the modern mobile era. And the JVM does not run on iOS devices.

Codename One is an open source cross-platform framework aimed at providing write once, run anywhere code for various mobile and desktop operating systems. In this episode, I interview Steve Hannah about the project.

[INTERVIEW]

[00:01:08] KP: Steve, welcome to Software Engineering Daily.

[00:01:11] SH: Glad to be here, Kyle.

[00:01:13] KP: So Steve, to warm things up, tell me a little bit about your background and how you became a software engineer.

[00:01:19] SH: Oh, well, I started out near the beginning of the Internet boom. Probably in the late 90s, I started making webpages. And then I started making a little bit of interactive webpages. Then I started making some CGI scripts with Perl. From there, it was just anything and everything. I'd buy a computer book or two every paycheck. I went to school at SFU. Did

computing science there. And worked at SFR for about 10 years doing mostly backend database work, PHP and MySQL stuff.

And then I decided I wanted to start making iPhone apps. And I started looking around. AI had a background in java at that point, and I kind of wanted to use Java if possible, and Codename One popped up. They had just come out. And they were sort of the only player on the scene at the time for making native iPhone apps. So I started playing around with it and started contributing to the project as it was open source. I mean, one thing led to another. They ended up hiring me. And that's what I'm here to talk about, I guess.

[00:02:23] KP: Well, it's a very cool entry point. For listeners who don't know about it yet, what is Codename One?

[00:02:28] SH: Codename One is a toolkit for building native mobile apps that are cross-platform using the Java ecosystem. So it supports Kotlin and Java. I've done things with other languages too. But primarily, right now, we support Kotlin and Java. Probably the closest thing is Flutter as far as if you wanted a comparison, it would be basically Flutter for the Java ecosystem instead of Dart.

[00:02:58] KP: So I know on the Android side, it's kind of native Java under the hood, or I guess Dalvik Java, if that's a thing, which maybe would be an interesting contrast of how you get it deployed there. But it seems like more straightforward deployment path. How do you get Java code running on an iPhone?

[00:03:13] SH: We've got a virtual machine that we wrote. But essentially, it compiles down to C code, and then we compile that C code using Xcode build, just all the standard Apple native tools, and that produces the app. At the very beginning we were using something called XMLVM. I don't know. Are you familiar with that?

[00:03:32] KP: I'm not. No.

[00:03:33] SH: That was a really innovative thing back it was available, I guess, 2012, 2013 that would convert code to sort of an intermediate XML format and then it could compile it down to

native C code. But we migrated away from that as it wasn't all that well-supported. It was an academic project. And we ended up writing our own virtual machine that we call ParparVM. And it's really quite simple. The power behind it is that it allows us to still use all the native Apple tool chains so that if we need to pull in anything that's like native third-party libraries or Apple changes the goal posts like they always do, it's fairly easy to keep in sync with it because we're working with actual Xcode projects at the native level.

[00:04:21] KP: And do I have to pay a significant penalty in my build size by including the VM?

[00:04:25] SH: Not significantly. I mean, the iPhone apps often come out smaller than the Android apps. I mean, there's a whole bunch of stuff that factors into how big the app is going to be, including resources, images, that sort of stuff. But the ParparVM includes the compiler portion that takes the Java byte codes and converts them into C. And it has a step that's kind of like Proguard that actually I was working on it today to help improve Kotlin support. That goes through and it prunes out anything that's not being used. So you don't end up with a whole bunch of stuff from the Java libraries that you're not using.

[00:05:01] KP: If I were committed to writing my app in node, I'd probably think about React Native. Well, I guess I should say if i was going to write my app in React, I would think about React Native as a potential tool similar to this for getting it on multi-device situations. Can you compare and contrast how your framework compares to theirs?

[00:05:18] SH: I don't have a lot of experience with React Native. You might be able to fill me in. As far as I understood, React Native, it uses the native components, but with the JavaScript wrapper around them. That's in contrast to the way Flutter does it, where they actually draw their own lightweight UI. Am I right about that with React Native?

[00:05:37] KP: Yeah, I believe so. I don't want to speak to it because I know there's a lot of facets of the project. But I think you've nailed some of the key ones.

[00:05:44] SH: Yeah. So we're closer to the way Flutter works, where we have a lightweight UI that's completely portable. It's drawn on the graphics stack of the platform. On iPhone, we use Metal, OpenGL. And on Android, we use their Canvas API. We can deploy to JavaScript also

and employ to the web. And in that case, we use the Canvas API. So what whatever a game graphic stack would be written on, that's what our widgets are based on.

In fact, for a long time, we were the only one in the space that was doing it this way notably in the mobile space especially. And with Flutter coming, they've been sort of the powerhouse that has taken a lot of the oxygen out of the ecosystem. But in some ways, it's been good for us just because it makes it easier to describe how it works. Flutter is doing it roughly the same way, just with a different language. It helps overcome some of the obstacles of doubters for a lightweight UI.

[00:06:43] KP: So Java is a language that's been around for quite some time. It's pretty mature. Has gone through some major enhancements. I don't know if you've seen any patterns in the people who are active Java developers or the types of companies that are still using it. Are there any trends like that that you've seen in the types of applications that are applicable for Codename One and where people are using it?

[00:07:04] SH: Well, it certainly is attractive to people that have a Java shop. They've got Java developers. So it's fairly easy to pick up. We've got our own APIs that are cross-platform to cover everything you would usually need to do on a mobile application. So that it appeals to Java developer because you can – It's like a Maven project. You download the starter template. You can build it in Maven and run it instantly in the simulator. So it'd be familiar you'd be using like IntelliJ. Well, I use IntelliJ. But you could be using NetBeans, or Eclipse, or anything that you can use Maven in. So it should be familiar in that respect. It's hard to generalize though. Certainly, historically, the API is similar to use as with Swing. I don't know. Do you remember Swing?

[00:07:52] KP: Yeah. Swing is still around. I believe they're still doing Spring One every year, popular web app framework.

[00:07:57] SH: Or Swing, swing like SW.

[00:08:00] KP: Oh, yeah, Swing. The layout formats. Yes. Sorry. It's been a while since I did any Swing, yes.

[00:08:07] SH: And swing actually is still around also. It was the desktop lightweight tool kit that Java had since just prior to 1.2. And so the API is very similar to that. At the beginning, that would be a very familiar thing, especially if you're a Swing developer, because a lot of the APIs were very, very similar to those. That's less of a drawing point now, because there's not as many people programming Swing anymore.

[00:08:31] KP: Well, the layouts in swing were something I really appreciated. Have you converted those over?

[00:08:35] SH: Yeah. Most of the key layouts were part of Codename One at the very beginning. The guys who created Codename One are Sun. They were Sun developers. And the project started out as an in-house project in Sun to make a lightweight UI tool kit that would work over all the different flavors of Java mobile edition. And that was a really nascent version of it though, and they ended up starting their own company once they expanded it to work with things like iPhone and Android.

[00:09:09] KP: Can I use Java 8 with your project if I want to have access to lambdas?

[00:09:13] SH: Yes. We do support Java 8. We haven't got like java 17. We're not up there, but we do support Lambdas. The runtime library that we support is a subset of Java 8. There are some things that you can't do, notably, Reflection, just because – Well, among other reasons, Reflection, it makes it hard to trim the app down and prune out the things that you don't need.

[00:09:37] KP: Gotcha. Yeah. So Java is famous for many reasons, but one being it's garbage collection. It was one of the earliest languages to have, I guess, a pretty strong garbage collection system. I don't imagine that's necessarily running on the devices, or is it? How does that translate into the different deployment settings?

[00:09:54] SH: On Android, it runs on the Android virtual machine. So it uses whatever garbage collection it has on there. On iOS, we have a garbage collector that we built into ParparVM that's optimized to not jitter. It's a fairly basic mark sweep garbage collector. We paid a lot of attention to making sure that it doesn't lock up the UI.

[00:10:18] KP: Neat. Well, one of the main attractions of a lot of applications being put onto our mobile devices is the sensors and things we can get there, from the microphone, to the camera, or the accelerometer and these kinds of things. What's the developer experience like if I want to access those components?

[00:10:34] SH: Well, there's certain things that we've got built into the core, like we've got a runtime library that standard with all Codename One applications, that gives you access to things like the microphone, and sound, and a little bit of the contact stuff, just the common things you would need. If you're starting to delve into, say, Bluetooth or accelerometer sensors, we don't have that built into the core, but we've got third-party libraries, and actually libraries maintained by us that you can use to access those.

And in certain cases, if you really need to use a specific low-level API that we haven't encapsulated, you can write native interfaces quite easily to incorporate whatever API you need in there, in which case, you'd be writing sort of a Java interface files for how it works. And then you'd be writing, for each platform, the low-level native code that interacts with the API's that you want.

[00:11:30] KP: So that's an interesting aspect of the project. You're presenting me one sort of interface, one framework toolkit that I can then deploy to both devices. But those device manufacturers are in no way trying to produce a similar API. And I think in some ways, they might be particularly different. Are there any challenges you have in providing a seamless experience?

[00:11:51] SH: Oh, yeah. Well, that's the challenge. When you're making a cross-platform toolkit, there are going to be challenges, because, I mean, Apple, especially, I'd say, has no interest in making their system easy to port applications to Android. They want you in their ecosystem. So, I mean, there will be some things that don't translate easily to an application. Like Apple is always coming out with these little extensions, features, like little snippets and stuff, things that will pop-up outside of your app, like on your home screen, rich push notifications. They'll often add sort of proprietary features that don't translate across very easily.

I'll say that background execution is always a bit of not a fun point for me, just because they both are constantly moving the goalposts and making more and more restrictions.

So trying to make a cross-platform API that will cooperate on both platforms and run the apps in the background has always been a bit of a challenge. But in these cases, what I usually do is I focus on how much I can do that is cross-platform. And Codename One is one of the few that you can produce an application that is literally one application, all your code is cross-platform, and you deploy it across devices.

And in the case where you need to do that refinements and get some native functionality that you just couldn't get cross platform, that's the extra 1%. Maybe I need to create a native API to do something, or I need to create a native extension and fit that into my app. But definitely the very proprietary native things that don't translate easily are the pain point.

[00:13:33] KP: And what are my options for either running my code live on a connected USB device or in some sort of simulator for both Android and iOS?

[00:13:42] SH: That's actually I think one of the strengths of Codename One, is it comes with a Java SE implementation. Like we have multiple ports. One of them is the Java SE implementation. And we've got a simulator based on that. We also use that same implementation for deploying to the desktop, because you can deploy your apps as a Windows app or a Mac app also.

So to run it in the simulator, I guess it's similar to Flutter. You would just press run in your IDE, and it pops up in the simulator pretty quickly. But we also have live reload that you can enable, so that as you're making changes to the code, it will relaunch the simulator without having to wait to build the whole thing again.

[00:14:23] KP: I know there's a release cycle. I see it myself more on the Android side. And there are then sunsetting older versions. I'm sure there's a similar process on the iOS side. What's it like? Or are there any challenges around managing that release cycle and how you bring new ideas into your own codebase?

[00:14:41] SH: Oh, yeah. Well, that's why there's never a dull day. There's never a we're done. Because every week, the Android or Apple announcing some new feature, or a new version. And with the new versions you're looking through, oftentimes they're removing things and restricting things. So it can be a difficult balancing act. Sometimes they'll make changes that you can just change the source code so that you'll support both old phones and new phones. But sometimes that's actually, well, either not possible or just more difficult. You actually have to make a hard choice and say, "Okay, well, we're not going to support devices older than X anymore, because in order to add this new functionality or comply with the new requirements they have on Apple, it would just be too hard.

[00:15:29] KP: So I know the project gives me the opportunity to write in Java or Kotlin. But there's a little bit more to it. Can you talk about the other pieces of the toolkit like the GUI builder?

[00:15:39] SH: Oh, yeah, there's actually a lot of different pieces. I mean, we talked about the runtime API. It's the library with all the classes and methods that you can use cross-platform that's got widgets, all the things you'd expect, like buttons, and text fields, and the pickers and those sorts of things. So that's the API, the widgets. We do have a GUI builder. Personally, I don't use GUI builders. But we do have it, that allows you to sort of lay out a form visually, and then it will compile down to an actual class that you can interact with the rest of your code.

We've also got CSS support for a lot of the styling, so that you can handle things like backgrounds and borders, and padding, and fonts, and stuff by using CSS instead of having to put that directly in your code. The other side of things that we do offer is, well, if you start with Codename One, you can build all yourself. Like it's an open source project. You can download the project and just start going. And if you've got a Mac, you can build it with Xcode, or build it with Android Studio. Like you've got a project you'd be working in, say, IntelliJ as your Java IDE. And then when you want to build your Android app, you would go build Android Gradle app. And what it does is it'll actually generate an Android Studio project that you can then build with Android Studio. Or if you want to build the Xcode app, what it'll do is it'll generate an Xcode project that you can then build using Xcode.

But one thing that we do offer is a build server that has a free level account so that you don't actually need a Mac to build for iOS or Windows to build Windows. You just hit build in IntelliJ, and it builds the project on our build server, and then sends back the resulting IPA file or Android app. So you don't have to deal with all the minutiae of maintaining the native tool change. So that's a fairly big part of what we offer also. Probably forgetting something. It's a fairly expensive project that covers a lot of stuff.

[00:17:50] KP: What does it take to get something set up where I can trigger the build server whenever I've merged a PR or something like that?

[00:17:57] SH: CI. Well, I do this with some of my stuff. You can use in GitHub actions. It'll just trigger a build, because the build server, we don't offer this for free accounts. It's for, I think, pro accounts or higher, just because it could end up taxing the build server quite a bit. So it's one of these features that we reserve for pro users. But you can automate a build so that the build runs, and then it sends back the zip file so you can just script that directly into your GitHub action.

[00:18:28] KP: Neat. And what's the story – Or I guess, where does Codename One end in my journey from idea to something live in the App Store? Can you help me with the deployment? Or is it just for artifact building?

[00:18:39] SH: We do have some extra stuff to help you with, say, generating the certificates. We've got an iOS certificate wizard, that makes it a little bit easier to generate your certificates and provisioning profile. But when you're actually submitting to the App Store, about the same process as if you were to build it yourself in Xcode. These are some areas that we're always looking for if there are pain points to help people with. It's sort of value-added things that we're interested in.

But I found that like you get your IPA file, for example, back from the build server, and then you just upload that to the iTunes Store. And you'll have to fill out all their fields and upload the screenshots and stuff like that. We do have some tools for generating the screenshots. But usually, I end up making my own screenshots anyways, because I want to be very specific about the screenshots I want in the App Store.

[00:19:31] KP: Well, if I'm going to get serious about my app as a product of some kind, I might want to do in-app purchases, which is got to be an extra challenge with Apple being certainly different from how one might do it on Android. Can your project help me in that area?

[00:19:45] SH: Yes, we do have an in-app purchase API. I wrote a tutorial that's like a three or four-parter a few years ago going over it. And I went back, I don't know, several months ago, and notice that most of the instant actions that I had written that involved the interaction with Google store or Apple Store were no longer all that helpful, because they change their interfaces in their backend so much.

There are some common aspects for in-app purchases that are the same across both platforms. And our in-app purchase API supports that. In-app purchases is still one of the hardest things to incorporate, from my point of view. To do it properly, if you want to have subscriptions, for example, then you need to have your own server that's pinging the backend for Google's web service and Apple's web service to check and see if, for example, the user has canceled the subscription, or if they've changed the subscription, and you have to sync it that way.

We've done, I think, about as much as you can for abstracting the differences, so that for the most part on the client side, you don't have to worry about the differences. It's just one API for handling purchases. On the server side, it's still painful. I mean, I've got some sample projects in Java that I've set up that I usually just go back to and do a copy and paste when I want to start with a server to keep track of subscriptions. But I wish it were easier.

[00:21:12] KP: Yeah, that may be out of your hands at that point. You'd mentioned there's a free tier, which is going to be great for a developer who wants to kick the tires. And maybe if they get really serious about it, they go to the pro tier. What's after that? What's the scope of services that Codename One handles?

[00:21:27] SH: I'll start at the bottom level, is like no tier. You don't need an account at all to use Codename One. It's an open source project. It's all on Maven. So I mean, you can go to start.codenameone.com, and it's just like start dot – There's a spring initializer thing. And you check a few boxes of what you want your starter app to be. And you hit download. And it'll be a

Maven project that you can open up in IntelliJ. And you can build it and it'll run right there in the simulator.

If you got a Mac, you can generate it as an Xcode project and then build it in Xcode. And if you've got Android Studio installed, then you can build the Android Studio project and build in an Android Studio. hat doesn't require any form of signup or anything. The next level is basic, which gives you access to our build servers, which is, frankly, quite handy. So you don't have to have Xcode or Android Studio installed. All you have to do is you hit build in IntelliJ. And it uses your free account to generate a build. And there's a limit to, first of all, the size of the application you can build with that. And there's a limit to the number of builds you can do per month. I can't remember what the limits are, but it says it on the pricing page.

The next step up is sort of a paid account, which gives you unlimited builds. So that's more or less with that. The next step up is the pro account. I can never remember what the prices are. But the paid account was somewhere in the neighborhood of like 20 bucks a month. The pro account is somewhere in the neighborhood of like 80 bucks a month. I can't remember exactly. And what the pro account gets you is – Actually, I should mention that we have support for a cross-platform push notification API also. So that you can just hit our push servers and it'll – There's an API to allow it to go out to all your devices, whether they'd be iOS or Android. When you get into the pro tier, it raises the quota on your push notifications to allow you to do it in a production setting. Whereas in basic, it'd be a very small – It's a lower quota.

And then when you get beyond that, we're into enterprise. And at enterprise is where we offer pretty much full package support. You can get phone support from us. You get your issues prioritize and feature additions prioritized also. It's the enterprise subscribers that really drive the direction of what gets built.

[00:23:49] KP: Could you expand on that? What are some of the ways you're being pushed? Or where's the momentum going?

[00:23:54] SH: Oh, well, if we've got enterprise subscribers that are like banks, which we have a few of, then they'll be very interested in features related to fingerprint authentication, which we do have an API for that, a cross-platform API. But if they've got very specific needs, then they

might want that API to be improved. And when we're deciding what we're going to do on the roadmap, it's sort of a combination of what's best for the whole community as far as making it a better product, but also with priorities on what our enterprise subscribers need.

There are also enterprise subscribers that are using these for POS systems, like terminal systems, that might have very unique needs if it's running on a specific type of Android tablet. It's not like the standard thing you'd get from your phone store. Then they might have some different needs that we have to develop for.

But sometimes it's just like there's new cool things. WebRTC was a recent thing that we added support for, and it was largely pushed because there were some enterprise subscribers that wanted it. As you know, in the space, Apple is always moving the goalposts. There's always changes. So if there's new API's coming out that an enterprise subscriber wants, then we'll prioritize it, as opposed to just somebody else says, “Hey, are you guys going to implement this?” Well, if it's a big job, and we don't have any enterprise backing for it, then it can be hard to really put it into the schedule as a priority.

[00:25:24] KP: Well, if I were the owner of that point of sale system you described, I probably want to have it branded and just my colors and styles and all that. But in other cases, I might want my app to look more native. What are the options for theming?

[00:25:37] SH: We do have a native theme for each of – The UI is still drawn by us. It's lightweight. So it works similar to the way a native theme would work in Java FX or Flutter. We make the widgets so that they look and behave like the platform. And it's not perfect, but it works pretty well. Usually, with myself, I'll all end up wanting just a unified theme across the whole thing. It ends up being easier to maintain in the long term. But yeah, definitely, in some cases, people just want that native theme. And that is when you first generate an app, there is an option to have the native theme. In which case, when you build the app for Android, it'll look like Android. And when you build it for iOS, it'll look like iOS.

[00:26:20] KP: So if I can Maven install just about anything, the world's the limit there. There're so many libraries out there I might tap into. Are there any gotchas I might bump into? Things that I want to install that are for some reason not transferable to mobile devices?

[00:26:35] SH: Yeah, I'd say in general, assume that third-party library that's not made for Codename One, like in Maven, is probably going to have some things that are incompatible. We've got a growing ecosystem of Codename One. We call them CN One libs. And we've got some that are on Maven already. We've done the Maven transition fairly recently. So they're not all in Maven. Actually, if you want to add third-party libraries, we have sort of like a control panel that you can go into the project settings, and you can search through the libraries you want to add, and you just press a button, install it, and it'll install the dependencies in your project. And there's quite a few in there now for most of the things.

So if it's a Maven dependency that's just doing some standard stuff, like say, some text processing, some math stuff and is sort of self-contained, then it will probably work. But if it's doing a lot of like networking, or it's got reflection, which a lot of third-party Maven libraries you're going to want to rely on, then it's going to run afoul of Codename One.

But usually what I'll do if I am interested, it's only a second to copy and paste into dependency, try it out. And it'll tell me right away when I run it in the simulator if it's going to fail the compliance check. And then if it does fail the compliance check, I'll look into maybe a different option or trying to port it into Codename One.

[00:28:03] KP: Could you give some examples of a few of those CN One libs? What are some of the libraries that have become available?

[00:28:09] SH: Well, there's lots. There's WebSockets. There's WebRTC. There's an accelerometer, CN One lib. Support for more of the traditional Java sockets API. There's a Google Maps one that allows you to integrate the Google Maps directly. There's quite a few. There's a couple that are for like the QR code scanning.

[00:28:32] KP: So if there's a couple, that must mean that these aren't all – That could be sanctioned, but they're not all coming from you guys. Is there a community of open source developers contributing as well?

[00:28:41] SH: There is. It's a small, but enthusiastic community, and they pitch in. We develop – Like you said, we do develop some of them. And especially if we've got an enterprise subscriber, or even a pro subscriber saying that they're having trouble with a third-party library, I'll end up downloading it myself and sometimes applying patches to that also.

[00:29:01] KP: So what's next for Codename One? Where's the project headed?

[00:29:04] SH: I think probably more in the Kotlin space. Kotlin seems to be the new rising star. And so there's a lot more developers that are using Kotlin. Always working on trying to improve the user experience. So there's improvements to the GUI builder. And also, I built a sort of a UI toolkit called CodeRAD. That is sort of a new way to build applications using XML. I always get bugged for XML as a new way. But using XML to define the user interface and more of a model view controller setup there. So there's a few things on the go, including just making sure we stay abreast with all the updates on iOS and Android.

[00:29:49] KP: Yeah, it seems like becoming the premier Java integration tool for deploying to mobile could be certainly work enough. But I'm curious if you have aspirations for new emerging devices. Like well Codename One run on VR things if we ever have them?

[00:30:04] SH: It's hard to say. VR things are off the radar right now. But things come on the radar pretty quickly if we see cool stuff or cool ways they can be applied. I'm a sucker for anytime I see a new platform and I'm like, "Hey, I wonder if we can get that working on Codename One." I mean, that's how I came to the project to begin with. It's really flexible. And it's lightweight, so that it can be portable. That's what is optimal for. So there's all kinds of possibilities. But yeah, the VR, not really on the radar right now.

[00:30:36] KP: Makes sense. Yeah. Well, as I was doing some background research, I came across one of your – I assume it's a side project. I'll ask you about that. In general, what is Tuxpin?

[00:30:46] SH: Oh, Tuxpin pin. I guess the background is that I discovered maybe a couple years ago that I really liked the audio medium, and more so than the video medium. I listen to a lot of podcasts. And I also, like in my daily schedule, what I would do is I wake up and I look

through my Twitter feed my news feed and I sometimes see articles that I want to read, and I just don't have time to read them. And so I'd add them to reading lists I would never get to. So I decided that, “Well, wouldn't it be cool if you could make a podcast out of the news stories that you're reading?” And the text to speech technology, especially the neural technology that's come out with – Well, Amazon's got that. Amazon Polly, are really good. So I tried hooking it up just like for myself to see what it would be like to convert like a news feed into a podcast so you can listen to what they're saying. And I found that I preferred listening to the Amazon Polly narration to like anything but a really good human narrator, because it's very consistent. And even if you get the odd word they get wrong, they get the same word wrong consistently the same way every time. And it's actually surprisingly rare that they get things wrong. It's just very pleasant to listen to.

So Tuxpin, I turned it into an app and a webservice so that essentially you can create a Tuxpin account. And there's a Chrome plugin, and an app that'll allow you to add any web page that you want narrated to your own private podcast feed. And so you add the Tuxpin podcast feed into your podcast app. And then it just shows up like another podcast. So I make my playlist in the morning. If I see an article that is too long that I don't want to read before I start my day, I'll just add it to my Tuxpin lists. Tuxpin will narrate it for me. And then when I go on a walk later that day, I'll listen to it.

So it's actually still under development. It's a fairly new side projects that I'm working on. The apps themselves for Android and iOS are built in Codename One, of course. There's a Chrome plugin that I just used raw JavaScript for. The backend is PHP. It actually uses a framework that I started building in 2005 called Zeta Phase for managing all the backend stuff. And it uses Amazon Polly for the narration. I'm working on something right now to try to make it so that like right now the app is directed at the end user, you could say, the guy that is doing what I'm doing. I just want to listen to whatever article I want. So I'm adding a feature for people that have, say, a blog, or if they run a newspaper, and they want to convert it into a podcast, like a narrated version. I'm adding that as a feature just to try to branch it out. I haven't really done any promotion for it. It's just a little side project that I really enjoy using. And I hope there's some likeminded people out there that like audio as much as me and they'll start using it to.

[00:33:43] KP: Absolutely. One of the features that kind of impressed me, you have it on your own blog, and I listened to one of the articles. And when I hit play, it didn't start by reading me the header and the menus and then the article. It really went to the title and then read me the article. How does it know where to start?

[00:33:58] SH: PHP library that's based on, I think, something that's sponsored by Mozilla. It's just called Readability. And I also do some preprocessing. But that's like the backbone of it that is it magically can look at a web page and figure out what's the cruft and what's the content. So it's almost 100% at knowing to not get the comments. Like, say, you would do a page that's got a whole bunch of comments at the end of it. You don't want all those. I mean, it'll take forever. I mean, usually you don't want. Maybe you do. But I don't want them. And it just gets the content of the article and that's it.

[00:34:33] KP: Very neat. Well, I'll be looking to see where the project develops to. I already found it useful listening to the article, and I can definitely see this popping up on other blogs.

[00:34:41] SH: Oh, cool. Yeah. Well, that's the part that I haven't released yet. You stumbled across it. I've only added it to the one story that's like the latest story that I wrote on my blog. I haven't added it to any of the other ones. And I haven't actually – It's still sort of – It's just there as my test page.

[00:34:58] KP: Very cool. Well, what's the getting started story for someone that wants to dig directly into Codename One?

[00:35:04] SH: I would go to start.codenameone.com and just press the – What is it? There's like a download button. It's got a few options like choose your IDE. It'll show you the sort of your sample Hello World project. And just hit the download button, and it'll download a zip file, that is a starter project. It's just a Hello World project. It's also got some links on that page to a tutorial that you can go through. Alternatively, and additionally, just go to codenameone.com. And there's a lot of tutorials. And there's the YouTube videos on how-to's and how this is all done.

One of the things that attracted me to the project originally was the support. How quickly – Even like not paying customers. Like you're having trouble post questions, and the founders at the

time, and still do, Shai, in particular. He can answer, I don't know, 100 emails a minute or something. I'm exaggerating. But they're very, very helpful. We try to be. So that's the place to get started. And hopefully, you can get off and get started and start making an app without any issues.

[00:36:05] KP: We'll link there in the show notes. Steve, thanks for coming on Software Engineering Daily.

[00:36:10] SH: Well, thanks so much for having me, Kyle.

[END]