EPISODE 472

[INTRODUCTION]

[0:00:00.9] JM: Most user interfaces that we interact with are not animated. We click on a button and a form blinks into view. We clink on a link and the page abruptly changes. On the other hand, when we interact with an application that has animations, we can feel the difference. The animations are often subtle and if you aren't sure what I'm talking about, you can pay attention the next time you use Slack or Facebook Messenger or iMessage. Airbnb values animations in their UI's so much that they built Lottie, a library for animation that we did a show about a while ago.

In an animated application, the user interface feels alive. When a software team takes the time to build animations into small interactions, the user perceives the animations as polish and attention to detail. Sarah Drasner has been evangelizing the value of animations for years and she's an expert at implementing complex and beautiful animations on the web. She works at Microsoft as a developer advocate and joins the show to talk about how to build animations.

If you are building a web application and you want to create a unique UI, then you might find this show useful. JavaScript supports detailed animations, often through the manipulation of SVG files. SVG stands for scalable vector graphics, a file format that represents an image in its own DOM. SVG is so flexible because of this DOM format, which defines the different parts of the SVG just like your webpage DOM defines all the different parts of a webpage. We know how easy a web page is to interact with.

This is in contrast to a bit map which is just a single matrix of dots without any rich meta data. So bitmaps are not very easy to manipulate programmatically. You could manipulate SVG with raw JavaScript but most people use a front end JavaScript framework like React, Angular or VueJS. Sarah has been implementing most of her recent web applications using VueJS and she's a member of the Vue core team. Vue actually has an entertaining story because it gained popularity at a time when google was supporting AngularJS and facebook was supporting ReactJS. The first version of Vue was created from scratch by a single developer Evan You. It's been quite interesting to see the rise of this framework that was just made by a single guy.

We actually did a show with Evan You a while ago, which you can look up in our archives and if you are a Vue developer and you're looking for an open source project to hack on, you can check out softwaredaily.com, which is an open source platform that we're building to consume content about software. In addition to the Vue web app at softwaredaily.com, we also have the Software Engineering Daily app for iOS and for Android and all of these apps are open sourced at github.com/softwareengineeringdaily.

If you're looking for an open source project to get involved with, we would love to get your help. Thanks for listening and let's get on with this episode.

[SPONSOR BREAK]

[0:03:20.3] JM: Women 2.0 is a company with a vision of gender equality in the tech world. Women 2.0 is a community, a media company and a jobs platform that connects top female talent with engineering jobs around the world. At the new Women 2.0 Jobs platform, find vetted jobs for women engineers, data scientists and product managers. To find a job that is right for you, go to women2.com/sedaily and if you're an engineering company, you can connect with top female talent on Women 2.0.

Companies like Twitter, Mongo DB and Craigslist use women 2.0 to find new hires. Go to women2.com/sedaily to find out how to post your company's jobs to Women 2.0. Thanks to Women 2.0 for being a new sponsor of Software Engineering Daily and check it out at women2.com/sedaily.

Thanks for listening and let's get back to the show.

[INTERVIEW]

[0:04:28.4] JM: Sarah Drazner is the author of *SVG Animations*, a book from O'Riley and she's also a senior developer advocate at Microsoft. Sarah, welcome to software engineering daily.

[0:04:37.7] SD: Thanks for having me.

[0:04:39.0] JM: I'd like to start with VueJS. Vue is a front end framework and I've worked with it personally, we have it on the softwaredaily.com website, it's a front end platform. So I've had firsthand experience with it and we did a show with Evan You about a year and a half ago, I think? It's been really easy, interesting to see the rise in popularity of Vue, especially in a market with Angular and React, which were too pre-established, very popular front end frameworks. What was it that Vue did differently to grow in popularity, in spite of the fact that there was Angular and React in the market.

[0:05:24.5] SD: Yeah, I mean, it's an excellent question. I personally haven't worked too much with Angular but I worked pretty heavily with React for a while and I still continue too, it's not for me — it's not for me it's not, you know, an either or, I think both are really pretty awesome.

I'm on the Vue core team now actually and I think some of the things that made it really exciting for me when I was learning it was that it basically seem to look over the entire landscape of all of the frameworks that came before it because Vue kind of came after and took all of the things that really worked well about these frameworks and then put them together in an elegant way. I think a lot of times when you end up taking the best pieces of a bunch of different disparate things, it ends up feeling very convoluted for the person working with it or using it.

That is not how it felt for me; it felt really, really wonderfully easy to work with, very intuitive. I think a huge part of it for me at least and part of the reason why I wanted to join the Vue core team was that the documentation was so good. The guide in the API docs are just like super fantastic. So I was really impressed with how well they communicated, what was available to developers and you know, got you going so easily.

[0:06:37.4] JM: I remember Evan You's goal of VueJS is building something that would allow for rapid prototyping. My sense is that the project has developed into something where people can use it beyond a rapid prototyping tool. They can use it for a complex project that scales.

Has it evolved beyond that rapid prototyping vision? Is that still the mission?

[0:07:03.2] SD: No, I think just like a lot of softwares that kind of like it comes out of needing a solution. I think even like React was born out of the Facebook team having this like pesky notification bug that they couldn't get rid off that part of the reason why it was appearing was because the state issue that they were having.

I think, you know, all of these frameworks kind of develop out of this need for something, right? Then they kind of expand into like obviously now React is not just for notification bugs, it's for like, giant large scale applications. Same with Vue; it was born out of a certain need that Evan had that wasn't being fulfilled at the time and now, it is suitable for tons of large scale applications.

There are giant applications especially in Asia like Alibaba and Y Bo. Also I know like Adobe and IBM and Microsoft are all like starting to do projects with that and adopting it. It's starting to get that like, you know, giant adoption in the West as well, for a good reason. Because it's very easy to maintain, it's easy to scale, in certain particular ways, that I really appreciate which is that if you are a maintainer on a project and a part of the way that I started working with you is I was a consultant and someone needed me to work on this Vue project.

Personally, when I heard that, I was like, "Oh now, I've got to learn another JavaScript framework. I feel like I just got react and now I got React and now I've got to go over this other thing and you know, I'm not proficient in this and I am proficient in React and I kind of took the contract with a bit of being tentative but, you know, some curiosity because I'm just kind of a curious person. I was really surprised at how easy it was for me to ramp up on what was going on, where everything was, it was super legible and things were really clear and clearly organized.

So I think in terms of being able to scale an application that a lot of developers can easily jump into and add to and understand where things are at or debug that was, for me, just a huge thing that I would just like, "Oh my god, I think I'm like super excited about this thing that I used to not understand what I had to do."

[0:09:20.6] JM: What do you think of the story of VueJS being able to gain so much traction? This is in light of React sponsored by Facebook, the multi hundred billion dollar company and

Angular, which was sponsored by Google, the multi hundred billion dollar company and then VueJS is just this one guy makes his own front end framework, you know supported by Patreon. I just, I can't get enough of that story.

[0:09:50.7] SD: Yeah, actually, it's not unprecedented in software development history that there be a project that's just kind of pioneered by somebody out of, you know, a need or you know, like having the benevolent dictator kind of person that comes in and says like, "I think what we really need is something like this," and building a whole community around it. I think there's actually some stories of it being really successful and I think this is one of them. What I really think is nice about it and, you know, I don't want to make the – I don't want to say like things that are negative about React or Angular because I think those are great projects too.

I'm excited about them being peers. I think when people talk about like framework wars, I'm never super down with that because I think that Vue stands on the shoulders of React in a lot of ways. I've also seen projects that came out of Vue that helped out React. Like React Transition group plus, which if we talk about animations later, we can dig into. But they learn from each other. I think that kind of competition between the frameworks and them learning from each other, just stands to help us as developers.

But in answer to your question, yeah, I think part of the reason that Vue is so successful in some ways, despite not having a giant corporate backing — certainly a giant corporate backing helps a project but also not needing to shift priorities based on a corporate schedule that might be just specifically for that project is really nice as well. A lot of Evan's decisions I feel like are really born out of his care for the framework. Again, not saying that other places aren't, but I've just seen a lot of things that he's decided that, you know, kind of came out of really thinking about the future of this framework and how best it can suit the most wide amount of people.

So it's kind of lovely to see and it's lovely to see the community embrace it, especially because with stuff like open source culture, that's the dream, right? The dream is that like, you don't have to have a ton of money in order to create something that is just fantastic that people can appreciate. So that part of like, you know, open source love in me is like, yeah, this is like the best possible scenario when you have a group of people that are just like passionate and

they're just like, "We're going to make this happen and then the community goes on Open Collective and Patreon and supports it and I think that that's really cool.

[0:12:19.9] JM: You spend a lot of time working with animations and tinkering as well as building larger scale projects that have a large animated component. Has Vue become your tool of choice for doing that tinkering?

[0:12:35.4] SD: Vue is particularly awesome at animations and again, I'll say that I have limited experience with angular so I can't make any statements there. Because Evan used to use frameworks for animation, I think he had some ideas and concepts about animation that were just really lovely and some of them, like I mentioned before, some of them actually have been ported back over to React.

Here's like a small example of something that I really appreciate. In Vue, you have this thing called transition modes. It's very common that when you have a component that's exiting and another one is entering at the same time. So let's say you have a list where somebody is bringing something in and something else is fading out at the same time. Inevitably, you're going to get into a situation where both of those nodes are in the DOM at the same time.

What ends up happening is that you get this like moment of [jank] or moment of – not [jank] but like visual inconsistency where their space for both of them and they snap out of place, even if they're transitioning and one of them doesn't seem like it's in the DOM and it is. You get this like, kind of clunky appearance. What you end up having to do in normal kind of imperative world is you either, if you're doing stuff with CSS, you'd have to chain things based on delays and that's very imperative and not very nice because if you change the length of the animation, you have to go back and find the timing function for the animation duration. So you have to go change that length of time as well. So you could make them connected but, you know, that ends up being kind of clunky.

Another thing that you could do is you can write a callback so that you're asking to, you know, "Once this is done, can you go find this other thing and make sure that that starts?" But transition modes end up allowing you to do this really declaratively, you can tell the component, "As soon as that one's done, go bring us that one in." That's done with one single attribute; you

just write “mode, out, in”. That’s it, that’s all you have to do to make those components come in totally seamlessly.

So that ends up being something that is like, as an animator or someone who, you know, focuses on animation or just as a developer working on an application where I end up having to re-implement the same code, to do the same thing over and over again, I don’t have to write that code anymore. That’s just something I can plug in to and I never end, it’s very common. So what ended up happening was, a Vue developer who was used to working with transition modes work – named cheap stake, had kind of been gotten used to that and really appreciated that about Vue and then had to work on a project in React and was like, “Wait, why don’t they have this?”

So he created React TransitionGroupPlus, which is basically like the ReactTransitionGroup that we are aware of and know but actually has that transition mode in it. This is like one of those examples of like, Vue did do a really cool thing here in terms of animation that maybe other people hadn’t thought about. But then, the community learned from that and now it’s all over the place, so that’s really cool. The other one that I think that is like super awesome in Vue is that transition group. Transition group component in Vue allows you to have animations that FLIP without doing any of the calculations for that and if you’re not familiar with FLIP, it stands for first, last, inverse, play.

Basically, what it does is it allows you to use transforms on things that are moving around in the DOM. The reason why that’s important is that when you’re using things like margined or top or left, it causes a lot of repaints for the browser, it’s very taxing for the browser and you end up getting a [jank] appearance where things are kind of sputtering across the screen. Transforms are the most performant way of moving those because they don’t cause those triggers, they don’t cause those repaints in the same way. You have this ability to hardware accelerate and it’s especially important when you have a lot of units to move around.

That tenant of first, last, inverse, play, what you’re doing is you’re calculating the distances using get bounding client wrecked. In SVG terms that would probably be getBBox and you’re finding the differences between where all of those things are on the screen and then you’re finding what that translates to and transforms. So when you write that code yourself and you have to do it a

fair amount, it's not my favorite code to write. I mean, you know, it's fine but it ends up being like a lot of, "Okay, go get this, go do that." Really, with Vue, you wrap things in that transition group component in your offered some hooks with which to observe when it's moving and when things are entering and exiting.

It becomes so easy to write all of that code, you don't have to write out exactly what's going on or anything. It just takes care of a lot of that stuff for you. So as an animator, yeah, those are like pretty big things, you know? This kind of limits the amount of the same logic that I have to keep writing again and again. I don't have to do that anymore and instead, can just focus on what the animation is doing and how it lives and how it behaves.

[SPONSOR MESSAGE]

[0:18:05.9] JM: Are you looking for a React or React Native job full time? Do you want to work with a Y Combinatory startup? G2i is a talent platform built for engineers, by engineers. Focused on react and React Native. I've personally hired engineers through their platform and the quality has been amazing. If you want a job in San Francisco that is React or React Native focused or you want to work remotely, go to g2i.co and signup now.

They offer both contract positions and full time positions. G2i is a platform or a React and React Native developers go to find work with the technologies that they love. Apply to work at G2i by going to g2i.co or you can send me an email to jeff@softwareengineeringdaily.com. I'd be happy to tell you more about my personal experience with G2i. I think you're going to love it, if you're looking for React or React Native employment full-time.

[INTERVIEW CONTINUED]

[0:19:14.7] JM: Let's take a step back to talk about animations more abstractly. The web is getting faster, our devices are getting faster, why don't we see more animations in the apps that we use?

[0:19:28.9] SD: I think because, you know, I deal with this a lot. I think it's because there's a couple of different things. One is that, when animation is done really well, it guides our user. I

think a lot of people have had experiences on Native where they can see that happening and they know that they don't know exactly that it's animation but they know that things are feeling very – They feel a little bit smoother every kind of interaction feels a little bit more intuitive. When animation is done poorly, it draws a lot of attention to itself. So I would say that the best animation to me is animation that doesn't call itself out to being animation.

When you're working with an interface, it's not like, "Hi, I'm animated." It's something that you don't — just feels a little bit more slick and you feel like you're a little bit more in control. Evolutionarily speaking, we are designed to notice something that's in motion both to protect ourselves in case someone, something is going to come eat us or to go eat something if we are looking for prey. So if something is moving around the page that doesn't need to move around on the page, it can be really irritating and I think everybody's seen this with like ads and stuff. Animation can get a bad rap.

People can experience something that is a poor implementation of an animation and be like, "Oh okay, animations shouldn't exist." But yet, when we, you know, experience really good animations and what I mean by this is if you go to something like an article like Paul Bakaus' Illusion of Speed, he talks about how a lot of short movements feel really long but when continuous motions feels very short. So if you employ animation to kind of smooth out those changes and the interpolations of state, it becomes a much better experience.

Another example I could use for this is like, you know, computers are used to bullions. Computers are like on, off, on, off. If something comes into our view, it could totally just say on and just be there. Humans aren't used – we're not evolutionarily designed to see that. If someone just appeared in your room before you without transitioning into your space, you'd be like, "What the hell?" So I think that that's like a big part of why animation ends up – that good animations ends up feeling really good to us is because it actually allows you to keep track of, "Oh, this came from over there. I understand spatially where that's coming from, where that's going to and where it's going to go in the future if I need to go get it again."

So I think one of the reasons why it's not taking as seriously as it should be or, you know, is designed into things as much as it should be is because people see a bad implementation of it

and they are scared off of it forever more. Instead of revisiting the fact that like, “That was a bad thing not a good thing.”

The other reason is, I think that people are not as aware of some of the tools that are out there that can help you. So that’s why working with Vue and I end up talking about green stock animation API a lot because I think it’s really amazing, it’s very declarative, it allows you to get a lot of work done very quickly and gives you a lot of power as an animator. If you pair things that are so amazing, like Vue and GSAP or GreenSock, it ends up not being so hard to implement. People will often be like, “Oh okay, I’ll just slap this on at the end.” Or like, “Oh, it was too hard to implement.” But if you use those two tools, it gives you a lot of control that even something like Native animation looks like and behaves like. So I think it’s because people are somewhat not aware of the possibilities that are offered to them, and I definitely try to put out resources to help people with that.

[0:23:14.4] JM: An engineering team has to put work into getting their stuff animated if they want it animated and we’ve already got teams that are doing design work and front end engineering and back end engineering. It seems like another engineering resource to allocate, if you want to get animations going. But we had a show recently with Airbnb about Lottie and they see animation as so valuable that they’re going to put significance resources into it Airbnb is kind of extreme when it comes to how much they care about UX and design.

On the other hand they’ve done really well. Are they crazy or is there a significant ROI for the companies who are investing in the animation?

[0:24:06.4] SD: No, I don’t think they’re crazy at all. I mean, I actually do think that it moves the needle. You see this in perceived performance of sites as well. I mean, think about If Apple or any of the Apple products that you used and no longer had any kind of animations or transitions in that interface at all. It will be a completely different experience and I don’t think it would be, you know, it would have caught on the way that it does. Because some of those animations are for people to understand where things are coming, where things are going and people talk about it in terms of delight, but I don’t even think it’s delight. I think it’s actually giving people the special awareness they need to exist in an artificial environment.

So if you're inviting someone to a virtual world, they do so with some trepidation, it's not what their brains are evolutionarily accustomed to. Animations ease that transition and make your interfaces a little bit more human and easier to work with. So some of that perceived performance is really important too. I know a lot of teams care about performance a lot and they should. Performance is huge, performance moves the needles. You can see how much money Amazon makes or loses based on hundreds milliseconds of improvement. It's enormous. But some of these is perceived performance too because, you humans over estimate passive wait times by something like 36% I think.

So what that means is that even if people, even if your console is telling you that something takes two seconds to load, it could feel to your user like it's taking a lot longer. If you have something like a loading state or something that tells you that something has been submitted and you're waiting or, you know, something to just easy along that journey, it might take the perceived wait time down from three seconds to one second or less. That's a really big thing. I mean, we do this in the real world too. If you look at like something like an airport, they actually make you walk, all the way around to get to your baggage for baggage claim, even though you don't need to do that because they want you to feel like your bags just appeared, you know, instead of getting you right there.

That is, you know, a change in perception. That's a perceived wait time. All of these things really make a difference and actually really move the needle. Some of them are kind of hard to measure though, some of them aren't exactly – Some things are easy to measure in terms of like how long people will wait on a page. I think in conference talks, I often show the study that [inaudible] did, were they used a custom loader instead of the kind of normal one that everybody uses that gif and they found that people were willing to wait twice as long for a custom loader. The custom loader wasn't anything really crazy or anything.

[0:26:49.8] JM: Like a custom loader wheel, like a cat running around or something?

[0:26:53.6] SD: It was just their logo with some dots on it. It wasn't, you know, anything totally insane or anything and people were willing to wait twice as long.

[0:27:02.2] JM: It wasn't the hour glass, it wasn't like the classic hour glass or the spinning color wheel on Apple.

[0:27:08.0] SD: Yeah, it was I think the one that they measured it against is just like those dots that are in a circle that you spin around?

[0:27:14.6] JM: Oh, right.

[0:27:17.2] SD: So yeah, I guess I feel very strongly that it actually does change business concerns.

[0:27:22.9] JM: Worthwhile. That said, let's dive in a little bit to how animations actually work, you've written a book on this. I think the place to start is the SVG file format. This is scalable vector graphics and there was a period of time where my little brother was obsessed with SVG's and I didn't really understand why. But then he showed me some stuff he had made with SVG's and I was like, "Well, that's actually incredible. It's beautiful stuff that you've made." I think the file format is at the root of this.

Let's say I have an SVG of a cat, it's an image of a cat. What does that file actually contain?

[0:27:59.8] SD: Yeah, this is the coolest part about SVG. When you're looking at a bitmap, let's start with bitmaps first, which people are more familiar with probably as JPEG's and PNG's. What they are is they're basically like a grid of like tiny little blocks of color, right? If you blow them up to be bigger and blow them like twice as big, you're going to get a loss of resolution, which we call lossy. SVG doesn't ever become lossy because unlike those bitmaps, what you have is basically, the equivalent of a piece of graph paper where you're plotting points and drawing lines between those points and you can either fill those lines with color or you can use the stroke of those lines.

So you're drawing with math, which makes it really amazing for things like charts and graphs. But it's also really good for images that you don't want to lose any resolution on. Like let's say a logo where on some pages you want to have a tiny one, and some pages you want like a big

splash graphic or something. You can blow it up to any size and it won't ever lose resolution. You are still sending the same amount of information.

So let's take the circle for instance. If you imagine you have this big piece of graph paper, it is actually endless. We have a graph paper that's endless in every single direction but in order to look inside of this piece of graph paper, we have a thing called the Vue box, which is basically like a little window that you are using to peer inside there. So then, for those of you who are familiar with the game Battleship, a lot of people know that game, you are saying things like, "C3," and then you would plot a point that you could see if the other person was like hitting something or something.

So if you were playing Battleship with SVG, you're basically plotting points along this piece of graph paper. You are saying, "Do this on the X axis and this on the Y axis, this on the X axis and this on the Y axis," and you make all sorts of shapes. So there's all these different elements and commands. If you look at a circle, basically all you're doing is saying, "CX, CY," and that makes a point on those two scales. Zo you are finding something on the X axis, something on the Y axis — a point — and then you say, "Radius," and that draws a full circle. So with three attributes, you now have a circle and the other thing about that is that unlike a Bitmap where let's say you have a picture of me and I am just sitting here and you want to move my arm around.

Well, in order to move my arm around, you'd have to go into something like Photoshop and go find my arm and cut a line around it and then you'd have to replace the background that is no longer there and then you have to absolutely position them on top of each other. That is kind of a nightmare. In SVG, all of those elements like what I just described for circle, those just exists on the DOM like regular DOM elements like a P tag or like an H1 tag. So you can slap a class on it, you can slap an idea on it and move those individual pieces around. So if you've got a hand with a bunch of fingers, you could say, "Finger one," slap a class on that and just tell CSS or JavaScript to move it and that's literally all you have to do to create these amazing animations with something that someone else has designed.

[0:31:17.7] JM: So what is scalable mean in this context? SVG, scalable vector graphics; what does the term scalable mean?

[0:31:24.1] SD: Scalable in those terms because we're drawing with math, because you can make things smaller or bigger without loss of resolution, that's what scalable is. That's scalable is in the name. So it is particularly awesome for our day and age of responsive. So if you have something that needs to be expanded and collapsed and work for all Vue ports, you don't need to do anything actually. You can set them at a particular width and height or you can take the width and height out of the SVG and it will just expand whatever the container is.

So you could use Grid, you could use Flexbox, you could let it expand to the entire width of the page. Really the sky is the limit. So that means that you don't have to actually change anything for responsive plus, plus! — This is where I get really excited, if you're moving things around inside of an SVG, inside of that coordinate system, inside of that graph paper, let's say you are moving a circle backwards and forward, it's using the coordinate system units to do so, not pixels. So that means that if you blow the SVG up or shrink it down, you don't actually have to use media queries or anything to change that animation. That animation is going to stay stable whether or not it is big or small. So you don't have to make any updates for responsive.

[0:32:40.2] JM: So that sounds like a very desirable set of attributes to have in an image file. But most of the images I engage with on the internet are JPEG or PNG and these images don't seem to scale as well. Why is that? Why aren't we using SVG all throughout the Internet?

[0:32:59.9] SD: Because it's not – I would say that there are cases for SVG and cases for not SVG. If you are trying to use a picture of a person, like let's say you have an "about the team" page. Those images of people's faces, you could make them an SVG but you'd probably end up having to do the same kind of work that a Bitmap does anyway, because you have to plot all of those pixels. So it may not be worthwhile to switch things over to SVG.

With that said, I know some tools like JPEG to SVG converters that can actually reduce the file sizes of PNG's and JPEG's by turning them into SVG's. Because you can use things like clipping areas where you are not coloring in all of those dots. So you actually, even for those files, if you change them into SVG sometimes, you could see giant performance benefits. That said, I think that a lot of line art and things like icons and logos and stuff is typically what people use SVG for. Any kind of vector graphics, or anything that looks a little bit more like flat or things

that are kind of drawn with a little bit more clarity rather than kind of fuzziness. Any kind of fuzziness or really rough sketchiness is probably better suited for a JPEG's and PNG's. That said, there are things like SVG filters and stuff that can create those effects.

So that's part of the reason why some people don't choose to do work with SVG but I think some of it is just the lack of education, to be honest. I think a lot of people reach for something like a PNG or a JPEG without realize that they could completely lower the cost and reap huge performance benefits by switching some graphics to SVG.

[SPONSOR MESSAGE]

[0:34:51.6] JM: You are building a Cloud Native application and you need to pick a cloud service provider. Maybe you are just starting out with a new app but you have dreams of scaling into the next giant unicorn. Maybe your business has been using on premise servers and you want to start moving some of your infrastructure to a secure cloud provider that you can trust. Maybe you're already in the cloud but you want to go multi cloud for added resilience.

IBM Cloud gives you all the tools you need to build Cloud Native applications. Use IBM Cloud container service to easily manage the deployment of your docker containers. For server less applications, use IBM Cloud Functions for low cost, event driven scalability. If you like to work with a fully managed platform as a service, IBM Cloud Foundry gives you a cloud operating system to control your distributed application. IBM Cloud is built on top of open source tools and it integrates with all of the third party services that you need to build, deploy, and manage your application.

So to start building with AI, IOT, data and mobile services today, go to softwareengineeringdaily.com/ibm and get started with countless tutorials and SDK's. You can start building apps for free and try numerous cloud services with no time restrictions. Try it out at softwareengineeringdaily.com/ibm. Thanks again to IBM for being a new sponsor. We really appreciate it.

[INTERVIEW CONTINUED]

[0:36:28.8] JM: Why is that? Why is it that it's cheaper to operate with an SVG? Because it sounds like SVG is much richer? I mean this is a file that has its own DOM associated with it. It sounds like a much more expensive file format. Why is it actually cheaper?

[0:36:44.8] SD: Because it actually doesn't take that much code to run it. So if you think about, okay, if I was going to draw a smile in a Bitmap and a smile in SVG, in SVG I have a circle, I have another circle and I have a path and that's three lines of code with very, very few attributes. In a Bitmap, I am plotting every single one of those pieces of the grid. So I have to use so many more, just to be super basic here, so many more ones and zeros to get that across. Just like I have so much more code to get to express that. So this is not to say that every SVG is going to be more performant than every PNG or JPEG. It's just that –

[0:37:27.9] JM: So you are saying SVG basically defines "here are a set of functions that draw shapes that are going to render in an image together". In contrast, a Bitmap is "here's a giant matrix of color and coordinates".

[0:37:43.9] SD: That's exactly right. So for that reason, JPEG's can be harder to manipulate. If you use something like Canvas, actually Canvas opens up a lot of possibilities. So like for Bitmap alterations and things, but you still run into the same problem with responsive. SVG is able to, in a smaller set of commands, draw things. It's built to draw things. That's literally it's job. So what I don't want to make it sound like is that you can just draw anything and it's cheaper.

Just like anything else, if you transfer a lot of data it costs more. So if you have an SVG that has a million bajillion things in it and a million bajillion path points, that's going to be expensive. So you're really thinking about it in terms of what the computer is delivering is really helpful but most of the time, if you optimize an SVG properly, it's probably going to be cheaper than PNG's and JPEG's and if you're curious on how to optimize SVG's properly, I wrote an article called *High Performance SVG's for CSS Tricks*.

That article goes through all the ways that you would output something and all of the ways that you can optimize something. I was pulled onto a project as a consultant where the load time for the site was 10 seconds long and just by properly optimizing their images, I brought it down to

under two seconds. So, you know, images are a huge way to reap benefits, performance benefits.

[0:39:10.7] JM: Yeah, you said images are two thirds of the bandwidth of the web and I think those are one of your talks. But why is it that poor image management leaves to poor web performance?

[0:39:22.7] SD: Because images are huge. If you think about how much like those giant hero images that people have when you visit their site and it is covering the whole thing and then you go down the page and there is a bunch of images, a bunch of pictures of things and stuff. So first of all that's a lot of HTP requests. Second of all, those images are a lot of data to transfer. Those giant images. So by thinking about images in a more responsible manner, you can bring down the size of your site by a lot.

It's really difficult to write by hand that much JavaScript and CSS as even a giant image can create. I think somebody said, I forget where the quote was from, but the really great quote was, "It's impossible to be a performance expert without also being an images expert," and I think that that is really true.

[0:40:18.4] JM: Getting back to the JavaScript libraries discussion, how does JavaScript interact with an SVG? What's the support there?

[0:40:29.6] SD: It's great, not really any issues actually. SVG, despite maybe some people's prior thinking — I think five years ago SVG wasn't really well supported. Now it's fully supported, we have Opera Mini, it's supported back to IE8. It's very well supported in JavaScript, it works really, really well with SVG's. In fact, I think people often think that I am using CSS for SVG's. I am almost never using CSS with SVG. I am pretty much always using JavaScript.

[0:40:58.4] JM: So let's say I am somebody who's – I am pretty good at Vue or React. Well let's just say Vue, and I'm a good coder and I have decided, you know, "I kind of want to become a little bit better at visual design and I want to integrate some animated elements into my personal webpage that's all built in Vue and so first, I want to build some SVG's myself. I want to go into some sort of visual editing tool and make these SVG's and then I want to animate them, and I

want to have these animations on my webpage, what am I going to need to do? What kind of software do I need? What procedures do I do to make these SVG art and then animate it?

[0:41:39.9] SD: Oh actually you don't need much. If you want to go on freepick.com or The Noun Project or whatever and go grab a free SVG, then you just take that SVG, you dump it inline into your page. So if you are using VUE, you could put it in a component or something then you, in order to animate it, you do that. You could use the same directives on all of the SVG elements. You can attach a class to it if you are using regular or IDE.

If you are using manilla JavaScript, if you are using Vue you would attach a raft and you can just animate that with methods. If you're going to have a transition if that's based on an entrance and exit, you could still use the transition component in Vue and plug into all of those JavaScript hooks that are available to use. So before enter, enter, before leave, leave, all of those will still work and be awesome and I typically use GreenSock for my animation needs. But there's tons that you can choose from and they all work really well.

Anime.JS is a really great one. Mo.JS is a super favorite of mine because you can spin up SVG elements inside just using the Mo.JS itself. You don't even have a graphic. You can just spin them up with JavaScript. So I actually think that JavaScript and especially Vue and SVG are just like amazing friends that people don't plug into as much as they maybe should even especially even because if you're not into – If you have trouble positioning things in CSS, sometimes SVG is really the answer. Just this last weekend, I was working on a project where I had a file loader and I have this plus sign in order for people to pick out the file from there. It was like an input type file kind of situation where they're gathering information from their own computer.

So I had this plus unit in CSS but I had to be for accessibility reasons, I had to make the plus and the P element be on the same line. So I had to use some Flexbox and it totally destroyed this plus thing. The plus was there and then it used to be a circle and then all of a sudden a scrunchy oval and it was a nightmare and no matter what I did with width and height, it was getting all squishy. So if I switch that to SVG which, I did which is just like writing that circle I told you about, writing a line and another line it's all of a sudden totally stable.

So if you are messing around even with text. Like if you want to do a text lockup that's responsive that works across browsers, throwing it in an SVG and never having to work with like worry about it ever again is pretty awesome. So it can solve all of your positioning woes in some ways.

[0:44:34.6] JM: What about if I wanted to do something more elaborate? Like if I really wanted to make my own images and animate them, I want to for example make a mailbox, a physical mailbox picture and then I want to have a mailman walk up to the mailbox and put an envelope inside the mailbox and I want to make all of these elements from scratch and then I want to animate them in the browser. What if I want to do something like that? Like how would I do that? What program would I use to create that art and then how do I bring that into the browser and animate it?

[0:45:09.9] SD: Okay. So yeah, I typically use Illustrator. I've been working with these graphic software for a long time so I know what the cool use Sketch and stuff. I found Sketch's output settings for SVG to be less than optimal and the person who works on the output settings for illustrator for SVG's is Dmitry Baranovski who wrote Snap.svg and Raphaël back in the days. So he is really smart about SVG's and did a pretty great job with the illustrator output setting. So I would say if you want to spare yourself a headache, you might want to use Illustrator but I understand if you want to work with Sketch like a lot of people do.

Inkscape is another great one, it's free. So the other two are paid for and Inkscape is not. So if you want to pay for graphics equipment, then that might be the solution for you. I haven't worked with that much because I just have the subscription to Illustrator. So I usually work with it in Illustrator. What I'll do is just draw a few things and then or bring in some other art and alter it and change it until it's what I need and then I make sure that I am labelling all of my paths. So if there is an arm, I group them together and call them "arm". If there is a head and I need to move it, I group everything together and call it "head". That really reduces the amount of time that you are spending looking at the DOM and trying to find out where everything is and stuff like that.

And those groups will be outputted in code. They'll be called G. So you can go find them and use them. If you label everything properly then you won't say "save as" in Illustrator, you'll say "export as" and when you say "export as" you say "SVG". It will give you a few options and save

it off. Then I go into this thing called SVG OMG by Jake Archibald. It's an SVG optimizer and that will allow me to change a few things and again, if you want to know more information on how to set all of these stuff up.

Or like I am speaking too fast, go to that high performance SVG post that I mentioned earlier because it details all of these stuff and it goes into a lot of detail about all the settings. Then after I've optimized it, I bring it into you know, directly in line in the HTML or into the component itself in Vue and then do everything I said for the last question.

[0:47:36.2] JM: Right, makes sense. Do people use SVG's for 3D models?

[0:47:41.3] SD: No, SVG doesn't have 3D capabilities.

[0:47:43.9] JM: Okay. I have seen these renderings you have on CodePen where it looks like a three dimensional situation. What are you doing with those?

[0:47:52.1] SD: I'm using three.js so that's not SVG.

[0:47:55.3] JM: Oh got it, okay cool.

[0:47:57.8] SD: Those are done with, what three.js is a way of working with WebGL.

[0:48:03.8] JM: Oh right, okay so I think I talked to somebody about React VR and they talked about three.js.

[0:48:10.4] SD: Yep, pretty sweet. It takes a little bit longer to get ramped up on three than it will for SVG for the listeners out there. But it's also really cool. It's really fun.

[0:48:19.1] JM: What's your prediction for how this 3D and the browser/VR stuff is going to play out?

[0:48:27.4] SD: I think I'm allowed to say this now because Josh Carpenter just did a talk on this so now it's public information but I had talked before working with Microsoft, I had talked

with Google about a potential contract that I turned down because of Microsoft about how they're integrating WebVR into the browser. So the future would be that some of the primitives that you see in A-Frame. Do you know what A-Frame is?

[0:48:51.1] JM: No.

[0:48:52.2] SD: Okay, so there is a thing called A-Frame, which gives you three dimensional primitives as HTML elements. So you would say, "A entity box," and then all of a sudden, a box is made for you and you have attributes that puts it backwards and forwards in space and stuff. So this A Frame thing is like really pretty cool for 3D and VR experiences that people can use without having a ton of, you know, like I mentioned three.js is a little bit more complicated set up. This is not. This is really, really easy to set up.

So the idea that Google is thinking about going with is to create this kind of A-Frame thing in the browser where these primitives are exposed to web developers and they basically are taking rather than responsive being this thing that goes sideways from device to device, like going small to large, responsive all of a sudden means a third dimension. The Z index. So creating those primitives that are available in the browser so that if you don't have 3D capability on whatever device you're on, it's a flat experience. But if you have a VR headset, it becomes a three dimensional experience.

[0:50:10.8] JM: All right, to wrap up, what changes do you think we're going to see on the web to support better animations, I mean, moving beyond the discussion of 3D of course.

[0:50:23.3] SD: Well, I mean, I think the 3D part is part of that. That change will be a change in animation too, they're also making animation primitive, which is part of the reason they were talking to me. That's going to be a big thing. I don't think that that's coming out in a year or something. I think that that's the future, like a longer look into the future and, you know, if the human race survives. I think in terms of animation, we have things coming out that are like web animation API, which I think is super awesome. It's not super well supported yet but there's some really bright minds like Rachel Neighbors and stuff working on that. Which allows the browser to extend those capabilities of everything that I was talking about in JavaScript like in GreenSock, it gives you some of that stuff natively which will be super dope.

So I think browsers are working on all of these things. I definitely see advancements in some of the rendering capabilities of every browser, I think edge has made some real leaps and bounds from where Internet Explorer was in terms of rendering and you know, Chrome is constantly pushing the needle on this so that's really exciting to see.

[0:51:31.9] JM: All right Sarah, well, I want to thank you for coming on Software Engineering Daily. It's been a pleasure talking to you and I enjoy seeing your CodePen animation tweets occasionally on Twitter. So it's always a nice little visual treat in the Twitter feed.

[0:51:47.9] SD: Thanks, thanks. Yeah, I have a lot of fun tweeting mine and other people's, especially if people are making cool stuff out there. So it's cool. Thanks for having me on, I really appreciated you taking time out of your day to talk to me today.

[0:51:59.6] JM: Absolutely.

[END OF INTERVIEW]

[0:52:04.3] JM: Do you have a product that is sold to software engineers? Are you looking to hire software engineers? Become a sponsor of Software Engineering Daily and support the show while getting your company into the ears of 24,000 developers around the world. Developers listen to Software Engineering Daily to find out about the latest strategies and tools for building software. Send me an email to find out more; jeff@softwareengineeringdaily.com.

The sponsors of Software Engineering Daily make this show possible, and I have enjoyed advertising for some of the brands that I personally love using in my software projects. If you're curious about becoming a sponsor, send me an email, or email your marketing director and tell them that they should send me an email. Jeff@softwareengineeringdaily.com. Thanks as always for listening and supporting the show and let's get on with the show.

[END]