EPISODE 925

[INTRODUCTION]

[0:00:00.3] JM: Data visualization is the presentation of data in a way that emphasizes certain qualities about that data. Data visualization can be used to prove a specific point, or it can be used as a depiction of a data set to be explored. Data visualization is used in consumer software products, as well as back-end engineering systems such as logging data.

As tools for data visualization have improved, data applications can consume more data at a faster pace. Browsers and mobile phones have improved, giving us the power to render high-fidelity, complex visualizations in near real-time and back-end systems and protocols have also improved, bridging the gap between the front-end and the back-end.

Sherman Wood and Chad Lumley are engineers working on Jaspersoft, an embedded analytics tool from TIBCO. Embedded analytics is a type of software that allows for the creation of data visualizations inside of an application. Sherman and Chad joined the show to discuss techniques for data visualization and how the field has evolved.

Full disclosure, TIBCO is a sponsor of Software Engineering Daily.

[INTERVIEW]

[0:01:16.4] JM: Chad Lumley and Sherman Wood, welcome to Software Engineering Daily.

[0:01:20.0] CL: Hi, thanks.

[0:01:21.2] JM: Today, we're talking about data visualization and data visualization is mostly made up of charts and graphs. We know how these charts and graphs relate to software engineering. We're always studying the analytics of our software systems. More recently, data visualization has also become important in the end-user application. You have consumers that want data visualizations from their finances, or from their fitness tracking software. There's a wider range of software that business people use, as well as consumers. You have more

companies becoming data-driven. How has the field of data visualization changed over the last decade?

[0:02:05.2] CL: With the latest developments and front-end design – I mean, computers are so fast right now that we can offload a lot of that back in technical stuff to actually on the front-end. It can handle a lot more, so we're able to put these more powerful charting applications in the front-end with more interactions. By the front-end, I mean the client-facing side of the application. It's more interactive now. There's lots of libraries available and able to interact with the data.

[0:02:33.2] JM: Does that apply to mobile, as well as desktop web?

[0:02:37.4] CL: Yeah. The client-facing part of the application, so – I mean, users can – basically, we're able to get – so mobile phones are more powerful now. Computers are more powerful now. We're able to put more powerful charting applications in front of the user and let them interact with it. That's a huge advantage.

Also, the way we handle data in the front-end is a lot more – we can do a lot more with data a lot, handle a lot more large transactions data. Yeah, definitely from an interaction standpoint, it's definitely a lot better than it used to be. Yeah, so does that answer your question?

[0:03:08.0] JM: Yeah. Can you go deeper on the data handling part of things? A while ago, we did a bunch of shows about the distinction between batch and streaming. We were mostly talking about back-end applications, but this is uniformly true across the application stack that streaming data is becoming easier to deal with. Not that we've stopped using batch, but can you tell me about what abstractions on the front-end provide better streaming interactivity?

[0:03:39.7] CL: Yeah. One form is Sockets.IO. Basically, that's just one method where we can receive this streaming data. We just open up a channel and just receive this really nicely formatted stream of real-time data. Then so now, we're letting users interact with this data that's real-time. They can see exactly – I mean, I use it every day to catch the bus. I need to see how fast I need to run to the bus. We can handle a lot more of these real-time interactions pretty nice.

[0:04:08.8] SW: Also, the widget tree in the applications, the actual visualization of that whatever that day that is is able to deal with those streams, right? It's not a typical old style request response type of thing. It's more you've got the widget tree is able to receive the stream and process it and generate the result, which for a long time there was not the way things were working. Also, people's expectations of speed and so on is higher. We've got better networks. It's 4 and 5G, rather than 2 or whatever, this other thing as well. All these things have changed the expectations of the consumer.

[0:04:55.4] JM: With widget tree, you're essentially talking about the API contract between the back-end developer and the front-end developer, I think, because I remember developing, like charting-based applications back in college. That was 2013, I think. I used some off-the-shelf charting libraries, or open source things. It was painful to fetch from APIs. It was painful to do batch processing, because you had to do a lot of coding on the front-end to as you said, to do that request response. Are you saying that that's moved into the abstraction of the front-end component where these things handle the data influx at a more – a fashion, it's a more harmonic relationship between the back-end and the front-end?

[0:05:46.4] SW: I think a couple of answers to that. One is the widget tree, the chart or whatever has got – you're able to just take a streaming data set and have that update the visualization in real-time. That's actually managed for you automatically once you get the connection basically correct and formatted. There is still always – what I find is there still always that – there's some data manipulation that needs to go on at some level. I don't know what would be one.

That the visualization needs a tree data structure to render its particular thing. I've got this streaming data set coming in, which is blip, blip, blip. I've got to take the individual element in the stream and apply that to the tree, so then the visualization updates. There's still that stuff needs to go on.

[0:06:42.1] JM: Chad, what are the other basic software building blocks that have improved data visualization?

[0:06:47.9] CL: Definitely these standardized front-end frameworks are good, because a lot of times the knowledge is out there. I mean, odds are somebody's already figured this out, how to do this if there's some problems. That's nice having these more standardized front-end frameworks to use, where people have already integrated in these solutions. That's really handy to have modern.

[0:07:11.8] JM: What are the prototypical challenges in the field of data visualization? What has not changed, despite the improvements?

[0:07:19.6] SW: When we started, it was some things about what's happening with data visualization. I talked about, I mentioned user experience. You still got to design this stuff. It has to make sense for the end-consumer. That is still there. There is still always the data mess that you've got. I've got a particular visualization that I want to – want to display how do I get – what data do I need to even get there? That may have to be developed and designed somewhere else. You often see microservice after microservice being created to support individual visualizations. All these things take time to develop.

What I'm often seeing, because of the way the roles that you have in in application development crew, right? Often, you'll see there'll be people who are focused on the front-end work, the actual JavaScript and single-page app and the JavaScript frameworks, all that side of things, and they'll have others who are focused on the data, or the back-end side. When you get that situation, you will find that the front-end people were waiting for back-end people to give them the data, this other thing, right? That is just the logistics of having to do these sorts of applications.

[0:08:39.1] CL: Yeah. From the UX/UI side, it's really just making useful visualizations that people can actually use, data that actually makes sense, which is a common problem. People just dump a bunch of data on the page and they're like, “Look.” It's not really actionable. It's not really – I mean, it shows you an overview, but it doesn't actually help you do anything. That's a one common pattern we see with applications. Me personally working with them is that that's why you need to apply that design to the visualizations.

[0:09:08.0] SW: Yeah. Look at your personas, right? Who are these people? What are they actually – what is their background? What information will they want to consume and what form should it take to get them to that right information, right person at the right time as quickly as possible type of thing? Rather than having to process, like look at a whole raft of numbers and that thing, which they have to go and dig through, right? That design thing is critical.

[0:09:37.9] JM: It's almost like a UX person's responsibility. What's the division of labor between a UX designer and a front-end designer that's actually coding these data visualizations? Obviously, there's some times where this is combined into one role. How have you seen that division of labor between designers and front-end developers?

[0:10:01.0] CL: Sure. I've seen it both ways, where it's one person does everything and where it's broken up. I mean, I prefer it where I have a designer. Me, the actual implementation of it, because it's good to have that back and forth. They design something that's interesting, visually appealing and makes sense to the users and then I have to basically translate that into the actual visualization. A lot of times, the interactions, they need to be adjusted. Because a lot of times, they can't think of every single scenario possible. I'm there to like, “Okay, this is how it's actually working. Now let's get – maybe we need to redesign this part.” It's definitely a back and forth process. It's not just like, “I make this, or I design this, you make it.” I think the best workflow is definitely to have it split up. I'm sure it's not for every case.

[0:10:54.7] JM: I've also heard – there was this this podcast I did a while ago with an investor. He was an investor that had written a book, called *Winning with Data*, it's Tomasz Tunguz. One of the things he talked about in that book was how when he worked at Google, there was this thing that he encountered called the data breadlines. That was basically whatever, 2001-2005, somewhere then. If you wanted to get data, like a nice little CSV file, or JSON file or whatever, you have to go to the data science team, or the data engineering team and say like, “Hey, can I have the nightly report of the advertising analytics?”

You're waiting in-line, because there's somebody that has to write a job to get you that data. This idea of the data breadline where was these impoverished front-end and middleware developers who had to request data from the data engineers from the back-end teams. Then once that data actually makes it to the middleware developers, the microservices developers,

then they have to talk to the front-end developers and then the front-end developers have to talk to the UX designers and it's just this bottleneck.

Over time, it feels like it's become more optional, that you have the option to consolidate some of those interactions into the more verticalized systems. I mean, I know Chad, you basically said that it's still preferable to have some breakdown, to have some division of responsibilities. What are the advantages to where we are 10, 15 years after the data breadlines and we can have – we can use these high-level, highly verticalized systems? Or how good are these highly verticalized systems? Is it at the point where a UX designer can essentially go to a self-serve data tap and build their entire dashboard end-to-end? Or do we still need some interaction from the developers, some interaction perhaps from the back-end people? What does it look like in the most abstract, low-friction scenario?

[0:13:04.9] SW: It depends on the scenario. That scenario you gave of the data breadline, that is pretty typical even today with a lot of the data science world, right? You've got this some group who's responsible for the analysis and the data wrangling and so on. The application, the UX side is saying, “Well, we want to see this.” Where is that? I don't know, we're going to have to go and sort that out, right?

Then I want that – once we've worked out that, then we need to get that on a on-demand basis, like let's click and we need to have it appear when the app needs it, when the consumer needs it sort of thing. Those finding the data and then making it are operationally available, can be still an incredible mess. I mean, what's the stat that you see on the data science side? It's like, well, the data scientist spends 60 or more percent of their time wrangling data, rather than doing data science, right? That's still a big hesitation.

In a lot of cases, when we – Jaspersoft deals with – a lot of people are putting visualizations on top of an application. The data has already been curated in a lot of ways. The visualizations may be using that data in maybe not originally consistent ways, or expected ways, but the data tends to be more available, and so it's less effort and you have less of a – less of that date of breadline thing going on. It's still a continual problem.

Then there's this thing of, “Well. Oh, I need to write –” Even if the data is there, I've still got to write a microservice to go on pull it into that front-end and stream it or whatever, up into where the consumer is actually working, or interacting with the system. There's still that mechanic. Get the data, wrangle it, make it all right, then then deliver it type of thing in an on-demand way. Still that overall process.

[0:15:19.6] JM: I'd like to get to talking about the available toolset for building data visualizations. First, I want to talk a little bit about examples. You both use plenty of software in your day-to-day lives, both as developers and as consumers. Can you give me a few examples of data visualization products that you use? Maybe things you purchase as a consumer, or maybe there's a banking application that particularly stands out to you as useful, or maybe you like some internal monitoring software. What are your best examples of data visualization products in your everyday life?

[0:16:00.1] CL: Well. I mean, my personally is Mint. I think, just really categorize all my spending and – budget is really important to me. I'm a stickler for it. I like going in there and being able to break it down into categories. Actually, I can go in there and manipulate my own data, which I think is very – that interaction with your data, I think is excellent. Being able to go in there and read, retag things and categorize. Because that's what a lot of the data wrangling is in data sciences categorization. I think now they have machine learning that does that stuff. That's example of probably my favorite right now, as far as visualizing data and allowing the users to have a good experience with it and being able to interact with it.

[0:16:43.5] SW: Jaspersoft has been helping applications to data visualization, either from straight document type, report type of things, as well as embedded visualizations in their apps for many years. Probably the best example of that is Waggl, who we worked with quite a few years ago. They provide a automated curriculum for schools who are doing the core curriculum in the US. They’re an independent startup. I think which started from McGraw-Hill. They came at us with a 60-page UX big, saying this is the dashboards that I want to give my students and the school administrators and the teachers and the parents about how the class or the student is actually performing.

They built that with – it was very specific look and feel, very guided experience through the whole set of ideas around how you look at the student performance. They have cute things, like the flock, pigs flying and that stuff, sort of cartoonified the UX for in all the metrics and so on to make it really simple and relevant for their end-audience.

They won a lot of awards. They went out of business for a while. They just come back and now they're part of Houghton Mifflin, the big educational publisher as well. I really love working on that. They were the first time that we really did deep embedding with Jaspersoft. I haven't seen a better process since.

[0:18:21.7] JM: That's a pretty interesting example, because you hear this term the consumerization of IT today, which is a very buzz wordy term, but there's a lot of truth to it. Basically, the idea is there are elements of Instagram and Snapchat and Twitter that to some extent, they hook you, right? The likes and the animations and little buttons and popping noises and tinkling noises and things like that. To some extent, they're actually – those are useful little tools, because they can – they can clue the user in that something is important, or something has changed. You can use them with varying degrees of subtlety.

Obviously, we've seen these things be used in a somewhat abusive fashion by some of the social products that really need to get you to click on ads and stuff. We look at something like Slack, or even Google Docs, really. You see usage of cartoony things, animation things that can really be productive. Part of the reason we've been able to do that is because of these improvements in abstraction, because we're no longer thinking about can I just get this basic, horrible-looking line graph to represent my uptime of my service? We've moved beyond that era. To can we make flying pigs and animations and stuff?

When we think about that the lineage of tools that we've been using to build these kinds of data visualizations and go from the era of I'm just thinking of the analytics and monitoring products that I used in my first couple jobs, pre mint.com charting and financial stuff. When we think about the lineage from those early data visualization building blocks to today, give me a perspective on how the tooling for building these data visualizations has evolved.

[0:20:36.0] SW: I was discussing this, or presenting this internally to some of our folks. Where did this all come from? I think it was driven by things, like the iPhone and that original – all that evolving expectation about what user experience should be, set an expectation with people about how they wanted to consume information. Alongside that, in parallel, what the timing was.

Google came out with material design, which had a really very broad and compelling design aesthetic, which included things like animation, use of color and so on, right? What's actually happened now is that people are expecting that's a rich experience. On the design side, it's the challenge for the designers to come along and actually provide that crisp information in a crisp way, an expected way, an accessible way all the way through it, right?

What you see now with applications is everything move to that more material design viewpoint. You look back five or more years, a web application which is doing something operational, look like a desktop app, right? No longer. You go into Workday nowadays, which is a leading software as a service, human resources, payroll and so on system. It's got a very, very different experience. It looks very close to that that what you're expecting on your Android or your iPhone experience there.

That's not quite answering the question on the tooling. With this progression, also as Chad started, the network, the machines in it and the devices on it have all gotten faster and faster, more capable. Again, things like, now we're doing everything in JavaScript. So much in JavaScript that we use not to – we'd have to do more on the server side or something like that. We can rely on the network and the processing power on the in device to give these rich experiences. A lot of it has been driven by [inaudible 0:23:01.1] single-page application sorts of things, which we've done in a very general, generic way. Then we started think about, "Oh, we've got this bootstrap thing. It's going to make everything responsive."

Then a alongside that work, there were frameworks like jQuery, Django, etc., that were providing these rich widget sets. Now there's just an explosion of different visualization libraries and so on. You can just inherit and plug in Deck.gl from Uber, which is data science mapping in your browser type of thing. It's been just a huge explosion of time, so a lot of the expectations are set by this evolving. We see visualization as some deals comes along and does the next thing, or makes it easy to use and develop and just keeps going from there.

[0:24:01.4] JM: What you alluded to there with Workday, a Workday is actually interesting example, because that's a old – I've never used – I don't use Workday, but that's a company that's doing really well. I would assume they've invested in resources in updating the UI. I mean, you see this thing with older companies that have stayed successful through the years. I think Zendesk is maybe another good example.

Other companies have not invested as much in refactoring their UI, but my sense is that it's becoming easier to refactor a legacy UI. I don't have any data to back that up, but do you guys have any perspective on what it takes to refactor and re-architect a UI these days? Is it getting significantly easier? What are those patterns for refactoring your data visualizations?

[0:24:52.0] CL: I don't think it's easier. I mean, it's still quite a bit of work. I mean, that's actually what a lot what I do in consulting is just a lot of companies just want to refresh. They just want to – it's worth the time to actually put in to completely refactor the front-end to their customers and to reach new customer bases. I think it's totally worth it for them, but it's definitely not easy to over – I mean, pretty much – I've had experience doing is just coming in and they're like, “This is what we had and we need something better.”

Then so you have the design pass, the designers will go through and analyze the application and distill it down, because over the years, a bunch of junk has been added to it. They'll actually distill it down to what it needs. I've seen so many dashboards and so many visualization software where it's just data just dumps. There's no interpretation of it at all. Then that's where the design comes in. You're going through and distilling this information. Okay, talk to the users. What do you want this to do? Then they'll actually sit down and talk to them. Then that's how you refactor.

I mean, it's not just completely one-to-one relationship between the apps. I mean, you want to make it useful. I think that's the trend right now is definitely doing that. I think it's a great trend. It makes people actually want to use their software, their jobs and stuff.

[0:26:15.2] SW: Yeah. I mean, with Jaspersoft, we're dealing a lot with people who have got some application, which has been around some what is it? I'm dealing with one large global

financial services organization that has got all this stuff sitting in MVC cobol that produces character-based reports and screens and that thing. It's like, we wanted to modernize this. It's like, "Okay." What's the design is the starting point?

Then it gets into the – as Chad was alluding to, well now we've got the design there. Have you got the resources to use modern technology, right? Often, I'm dealing with a lot of people who are moving – take going from an on-prem deployment, which maybe running on an enterprises – within their data center type of thing. It's all got to go to the cloud. It's like, well, there are the same technical choices that you had that worked on the on-prem stuff going to work in a cloud? Probably not. Oh, and there's this thing called containers. What does that all mean? Or AWS and I can use all these services. That's lovely. Then of course with it all, these experience and the data visualizations on the side moves along that too.

[0:27:31.3] JM: You've mentioned something a couple times, embedded data visualization. That is a technique that contrasts with the use of general data visualization libraries. Can you describe the difference between embedded data visualization and general data visualization libraries?

[0:27:49.5] SW: If we talk about on the Jaspersoft end, we talk about embedding the visualizations, because typically for historic business intelligence types of tools, they provide a complete user experience. You go into the tool and give you all the dashboards and navigation and all the reports and all that stuff. It was all within that tool and this is part of the reason why people – There wasn't all broad use, or adoption of business intelligence across, say an organization, because the information was not where people were experiencing it.

What we've been doing on the Jaspersoft and we coined the term embedded BI, embedded data visualizations in Slack? No. You want to make the visualizations that are coming out of this rich set of data, which is curated and then be able to present visualizations on top of that, in a portal, or an application right where people are wanting to experience it. They want to make it interactive. You want to give the developer control to move on a workflow across the different visualizations, so that it's consistent and makes sense for in terms of user experience. That's the shift that we've been doing.

[0:29:17.5] JM: In a data visualization library, there are different ways that data can get ingested by the system. Are there different data ingestion patterns for an embedded data visualization, versus just a general UI tool?

[0:29:34.0] SW: Well, just quickly, there are the two forms of it. You can do as you were talking about originally, Jeff, about the – you had to get the data from somewhere and you bind it all into the visualization and then you – on the server side and then you throw it to the client, right? That's server-side rendering and data wrangling and all that thing. That's old-school. It tends not to be the way anymore. To typically, you've got the visualization machinery now running in the front-end and the back-end gives the right data to that visualization, right? Then the visualization process it and renders it up, but that could be that streaming data or whatever. There's more smarts in the front-end nowadays. That's of really two modes that I see happening there.

[0:30:30.4] CL: Yeah. Definitely in the front-end, it's more capable now of handling any – you can just hand the front-end a data set now. Pretty large one. They actually even have front-end databases that you can use, where you can actually just store tons of data and manipulate it. It's a lot easier now for the front-end to be able to actually not rely on the back-end to do a lot of the data grouping and organization and stuff. You can actually on the front-end. It's not the best thing to do, but it can do it. If you have if you have an application that needs to be very interactive, you can handle all that in the front-end. You don't need to just make a call to the backend, have the backend do your operation, send it back and then you reload the front-end. You can actually do that all the front-end now. Computers are really powerful, especially – and phones especially. Phones can do a lot.

[0:31:18.8] SW: Some of the Deck.gl demo is like, “Let's throw a million data points at the visualization running in the browser.” It's drawing away on the canvas like crazy and doing lots of stuff up there. Yeah.

[0:31:31.4] JM: You've mentioned that Deck.gl project a couple times. Can you describe that in more detail? You said it came out at Uber?

[0:31:38.0] SW: A lot of the larger Internet or software Internet-based companies, Uber has got a variety of visualizations that they open source. What is it? I'm looking at Airbnb originated

protocols, which is now called super set and that's just about to go under $1 with the Apache Software Foundation. That's a complete data access and data visualization platform, that's going to be nice and open source, right? It can plug in this vast variety of different open source JavaScript libraries for the actual visualization.

Deck.gl was, I think just seeing the other day that it got react views and other visualization library. Deck.gl, because it was all about – all that vehicles running around the place and that's what the thing, this deck.gl is a very sophisticated mapping visualization library with lots of – a lot of the analytics on the actual mapping data is actually happening within the framework that's going to be running in your browser, rather from the server, the server side, the back-end generating all the analytics and the front-end just rendering it. That's another thing, you see. There's a lot more capabilities in these various JavaScript libraries to do, not just data manipulation, but a lot of real analytic work. Tensorflow. Tensorflow in your browser, that thing, right? Lots of fun up there.

[0:33:12.8] JM: Let's come back to the topic of embedded visualization. Can you talk about some common patterns and use cases for embedded visualization?

[0:33:22.8] SW: Yeah. Typically, back to that comment about material design, there's a lot of expectations about the way an application should look and interact and feel, vary from the pure data visualization and you see a lot of very dashboard style of UIs, you see multiple elements, which are driven by parameters. You update the parameter and they – all the elements on the page change, or this navigational flow. I click on one thing and that changes something else in the screen and so on.

All those individual visualizations need to be created somehow and then all wired together, right? Now that's a one style. A lot of people want to try and do all the dashboards together. Dashboard is like a single object. We do that at Jaspersoft a bit. A lot of time, it's up to the web developer to build the individual visualizations and then wire them all up. Another thing that we see a lot, what people use Jaspersoft quite a bit is it's not just – the data visualization is like a chart on the screen, but people want to take data away from the app. We have a whole reporting thing as well.

I click on something and I get a PDF, or an excel, or something like that is also something that that people want to get. Probably the last thing that is really like the – we see as a best practice is it's a variation of what Chad was saying about Mint. He's able to go into Mint and go and change some things and that actually is part of the application that you've now tagged stuff up, right? When you've delivered an application to an audience, they often have more and evolving information needs.

If you've got a set of visualizations up there, you can bet your bottom dollar that people will be asking for more, okay. How do you create more? Is that a development task? Do the requirements and then create new pages and different workflows and how to get to those, that information and so on. Or, do you give, or do you think about self-service? What do you want to give the audience to serve themselves in the context of your application?

In Jaspersoft, we have a in-browser drag-and-drop way to create a – just make charts and cross-tabs and maybe just a listing of data, guided curated data experience behind the scenes, people can satisfy their own needs. Oh, I can go and create this visualization, I could stick that in a dashboard. I can run that as a report and have it downstream for other processes or myself. We see that quite a bit, that range of different styles of embedding right now.

[0:36:25.7] CL: Yeah. Embedding the application, it was great for the front-end, because I can just stick in – it's where the chart actually goes, or where the visualization will go on the page and this is actually managed elsewhere. That's what the Jaspersoft stuff does. It's nice, because I could just stick something on the page and allow the business analyst and stuff. They can take care of the visualizations. It's almost putting a placeholder. It takes a little pressure off of me and also it helps them keep the visualizations modified, up-to-date, and they can change it whenever they want to from whatever platform they're using the bag. That's just for what I've seen from embedded applications.

[0:37:07.4] JM: Jaspersoft is an embedded visualization tool that you've both worked on. Describe the spec for building an embedded visualization tool. What does it have to be able to do and how do you go about architecting a tool like that?

[0:37:23.4] SW: I was the architect originally, quite a few years ago now. I can probably talk to that the best. First off, you got to – the platform is got to do quite a few things. You want to be able to do things, like connect to data and have that managed in a secure way, for example. You want to then have a way of obviously, have that range of data visualizations available to you, so that people can just consume it.

Then it's all about, what are the APIs into that environment? I can say from the outside, an application of consumer and the application can say, give me this visualization and the actual infrastructure goes all the way through and can render that up to where they need it. You need to be quite an array, vary API. Sometimes it's just – it could be just simple rest calls, or something like that. It could be, give me a complete dashboard, a complete set of visualizations as a logical whole and include that within my front-end.

Then it's all things about security around when individuals interacting with the visualizations and the data, they should only be able to see and do the things that you want them to. The platform needs to be aware of who they are and its rules in place to control data access and control the visualization, so that they're not going outside their guardrails. Then of course, it's also scalability and so on. You've got maybe quite a large audience all-demanding. You've got to have the platform be able to scale up to allow all that to happen at once.

[0:39:16.3] CL: Yeah. From the front-end perspective, it's always nice to have a good JavaScript interface written in pure JavaScript, maintains all its dependencies, so I'm not going to have to handle dependencies. Just have the self-contained little interface that has all the API, has its own API that I don't have to mess with interacting directly with whatever the service is.

It's nice to have that layer. That's the way I would architect the front-end portion of it is you just have this nice interface that I can use in the front-end to call the back-end and not have to worry about dependencies and not have to worry about bringing in styles for the charts and stuff like that. Everything maintains itself.

[0:39:58.4] JM: How does an embedded visualization library work alongside other front-end libraries? If I'm using embedded visualization, how is it integrating with React? What's the

division of responsibility between what I'm building with an embedded visualization tool and what I would build with stand-alone react components? For example, if I wanted to use that open source library from Uber that you mentioned earlier.

[0:40:23.2] CL: I guess with the React side, so if you're just using an embedded application, I wouldn't say it's easier. Using it with React really doesn't depend on React. It's really just me putting a visualization in the application. What's cool is that you can actually – with Jaspersoft and – you can actually just use the data instead of the actual visualization and actually plug in your own type of tool. Then you can use React to handle the data on our side and then use custom visualizations and stuff. React is good for data management. If use it in conjunction with this Redux, which is a way you handle a state in the application. Yeah, does that answer the question? I'm not sure.

[0:41:08.6] JM: Yeah, sure. Sherman, would you elaborate on that at all?

[0:41:12.3] SW: Yeah. I mean, I think to Chad's point, we can use – we do the embedding in a couple different ways. You can, as Chad was saying, just use Jaspersoft as a data service, these APIs to basically get list values, or JSON structures or whatever. Then you can use whatever JavaScript front-end, like a deck.gl. Or we did some things with the TIBCO mapping service called TIBCO GL Analytics, to sit on top of data that was coming out of Jaspersoft and just use the native – first native APIs.

The other bit of it is Jaspersoft, its main thrust is you define the data visualizations using the Jaspersoft toolset. You have them sitting now in within the Jaspersoft service behind the scenes. Then you use JavaScript, the JavaScript library, visualize.js, or rest calls to go and retrieve the visualization and include them in the page as single units, which can then be made interacted by the web developer.

In that the visualization generation side is server-side rendering, basically. Then it's a relatively simple API to actually include them into the React app. It's a few lines of JavaScript. Rather than hundreds of lines of D3, or something like that. That's all a lot simpler and more productive for the web developer to go and get these, to have these interactive visualizations happen in a

controlled manner, well-secured enough of things, compared to writing it all by hand in JavaScript and microservices on the back-end and that thing.

[0:43:06.0] CL: Yeah. I think that's the main advantage, it's just a plug-and-play type where you just – I don't have to mess with the visualization part. I can just put the container on the page and interact with the API and then that handles the way it looks and they can handle that and I can. That's the advantage of having an embedded visualization like that is that it's taking a lot of load off the front-end for you. Just another approach.

[0:43:29.1] JM: Sherman, you've been working on Jaspersoft and embedded visualizations for a pretty long time. We've also talked about – throughout this conversation, we've talked a lot about the evolution of the developer, of the changes in the data visualization libraries, the changes in the requirements for data visualization, the improved user interfaces and the changing requirements and expectations for users on that side. How has the architecture of Jaspersoft and the resultant, or I guess the product requirements for Jaspersoft, how has the project evolved since you started working on it?

[0:44:10.6] SW: It has been a while. When we started the Jasper report server project in the mid-2000s, it was – because it was a Java-based platform that was much more around the programming language side. It's open source in Java. We're doing the same things as the other business sellers as vendors, like business objects or whatever. We had our portal that people logged into and they just experienced – that user experience just through whatever UX was available within Jaspersoft, right? It always had APIs and that you could do different things. You could do some limited embedding with an iframe. It was still Jaspersoft running it through it all. Now about five years ago, because we'd already been doing embedded work with reports and other things. We had our APIs and so on. It was like, "Oh, well we can see this. Everything's going single-page app now." Developers want applications, want this richer, more fine-grain approach to experiencing the visualizations that we can do in Jaspersoft. That's when we went down to creator in JavaScript library, visualize.js, that you could just plug in and get access to all that same rich visualization power that you've got within the Jaspersoft environment.

It has evolved a lot. I mean, now we still have a web user interface. I'd say that most of our customers and use that for say, administrative purposes. All the rest of the experiences being –

has happened either embedded, or batch style production reporting, or something like that. There's nobody directly interacting with that service, except for [inaudible 0:46:11.0] purposes, sort of thing. That's been an evolution there.

The other thing that we've seen over time is the –it's been the technologies, like single-page applications, or we're doing things with node, for example JavaScript on the back-end too. How do you fit this Jaspersoft platform into that? We can run in containers and that thing. It still is all Java-based, but people only interact inside an API, so we now – our renewal container world, or your multiple virtual machines, or whatever. In Jaspersoft, we say we wanted a couple of those within your environments and stuff. That architectural side move the cloud is changing a lot of things as well.

[0:47:05.1] JM: What's your vision for how the product looks five years into the future?

[0:47:11.2] SW: Well, that's the thing I've been banging on about for a while. I'd have to say that some of those, the base administrative and development UIs that we have are getting long in the tooth. They're very powerful and more sort of thing, but we need to keep up with that. It needs to go to much more of that material design view. That's more purely on the internal side of Jaspersoft really. I don't know, whether we'll probably get richer and easier data visualization creation tools around it. We want to reduce the amount of JavaScript work that people are doing to get to the visualizations that we need, so we're continuing to improve the through those tools there.

At the moment, you can create any visualization you want in Jaspersoft. I don't see us better and better APIs, richer, more flexible better ways to just access data if you need to go and do it inside your app. Those are the things that I would see us working on.

[0:48:19.6] JM: All right, final question for both of you. Do you have any other software trends in mind that you think will intersect with data visualization in impactful ways, or things that other people might not have thought of?

[0:48:34.8] CL: Well, from a front-end perspective the new trend everybody's going towards is these progressive web apps. Basically, it's the front-end is turning into like a native application,

like a web application is turning into really more of a native type layout. Because with progressive web apps, you could – they actually function offline. If you had application with charts and stuff and you went to the website and you're offline, or maybe got cut offline, you wouldn't lose the site. You could still see your data. It wouldn't be up-to-date if it's real-time data, but you could still see it and interact with it, even though you weren't connected to the Internet.

See a lot of trends in that direction coming up very soon. It's progressive. There's definitely a movement going on right now, where everybody's starting to gear their apps towards that, because it's a much better experience for the user. Also, just using more of these ugly interfaces that you see just disappearing and turning into something nice and pretty, I think as companies realize that they need to think of their customers first and how people are using their software, as the same thing with data, especially with data visualization stuff. That's the trend I see. I hope I see the redesign and stuff, but we'll definitely see a lot more of that. It is happening right now.

[0:49:53.4] SW: I would say that a lot of the apps that I see with the data visualizations in them, in some ways they're relatively straightforward. Because you're not trying to confuse people or anything like that, so you don't see a lot of different visualization types, or extreme visual – you don't really see infographics like you would expect out of from a newspaper or something like that, being presented inside apps. That's all down to your audience, basically. You want to get that right information, right person at the right time, sort of thing, and want to make that as easily consumed as possible.

I don't see a lot of data visualization changes from that perspective. I do see as we go more and more down for machine learning route, there'll be much more, much richer analytics being presented. It'll be easy and easy to get to. It's often a competitive advantage for applications that they're now able to get richer information, because this may be automated curation of the data coming along and then being exposed through the data visualizations. That's probably where I would see more and it's just easy and easy to hook into that AI/ML world and add value to the visualizations that people are actually interacting with.

[0:51:20.6] JM: Okay, guys. Well, it's been a real pleasure talking to you. I want to thank you for coming on Software Engineering Daily.

[0:51:25.6] SW: Thanks a lot, Jeff.

[0:51:26.3] CL: Yeah. Great time.

[END]