EPISODE 1106

[INTRODUCTION]

[00:00:00] JM: Users do not use web applications in a way that you might expect, and it's not easy to get the data that is necessary to get that full picture of the strange ways that your users are probably using your application. But a newer API within browsers does make this more possible by capturing DOM mutation. The change capture of these DOM mutations can be stored for replay in the future. And after being stored, this change capture can be retrieved and replayed. And this allows for a comprehensive frontend monitoring, which has been built into a product called FullStory.

We also covered a previous company called LogRocket, which does something similar. So if you're interested in the comparison between those two, you can go back and listen to that episode and also listen to this episode.

Michael Morrissey is the CTO of FullStory, and he joins the show to talk about this full session capture technology, the architecture of the company. How sessions get saved, and stored, and retrieved? And it's an interesting comparison to that previous episode we did.

If you're looking for all of our content, you can subscribe to the podcast, Software Daily. And if you're interested in becoming a paid subscriber, you can go to softwaredaily.com and subscribe. You can get ad-free episodes. With that, let's get on to today’s show.

[SPONSOR MESSAGE]

[00:01:29] JM: You probably do not enjoy searching for a job. Engineers don’t like sacrificing their time to do phone screens, and we don’t like doing whiteboard problems and working on tedious take home projects. Everyone knows the software hiring process is not perfect. But what’s the alternative? Triplebyte is the alternative.

Triplebyte is a platform for finding a great software job faster. Triplebyte works with 400+ tech companies, including Dropbox, Adobe, Coursera and Cruise Automation. Triplebyte improves the hiring process by saving you time and fast-tracking you to final interviews. At triplebyte.com/sedaily, you can start your process by taking a quiz, and after the quiz you get interviewed by Triplebyte if you pass that quiz. If you pass that interview, you make it straight to multiple onsite interviews. If you take a job, you get an additional $1,000 signing bonus from Triplebyte because you use the link triplebyte.com/sedaily.

That $1,000 is nice, but you might be making much more since those multiple onsite interviews would put you in a great position to potentially get multiple offers, and then you could figure out what your salary actually should be. Triplebyte does not look at candidate's backgrounds, like resumes and where they've worked and where they went to school. Triplebyte only cares about whether someone can code. So I'm a huge fan of that aspect of their model. This means that they work with lots of people from nontraditional and unusual backgrounds.

To get started, just go to triplebyte.com/sedaily and take a quiz to get started. There's very little risk and you might find yourself in a great position getting multiple onsite interviews from just one quiz and a Triplebyte interview. Go to triplebyte.com/sedaily to try it out.

Thank you to Triplebyte.

[INTERVIEW]

[00:03:46] JM: Michael, welcome to the show.

[00:03:48] MM: Thanks very for much for having me, Jeff.

[00:03:49] JM: I'd like to start with a discussion of frontend bugs. So when a bug makes it to the frontend, what does that bug typically manifest as? What are the issues that a user might encounter if there are bugs on the frontend?

[00:04:06] MM: Well, a lot of different things, namely confusion and frustration are two of the most common ones. We're certainly living in an age where all of us are relying on the Internet and applications more and more for the vast majority of our lives. And so we kind of expect these experiences online to be as perfect as possible. And when they're not, particularly if you're in a rush or if it's for something important, it can be really frustrating for users.

[00:04:38] JM: And how does it manifest for the developer? So the developer maybe becomes aware that users are having negative consequences of bugs being shipped. How does the developer become aware of those bugs?

[00:04:53] MM: There are a few different ways. Unfortunately, usually it takes – There is a long circuitous path before developers find out about issues in production, unless they are sort of really large, very sort of large-scale issues that you would notice in monitoring graphs. The problem is we all think we understand what the users are actually experiencing with their product. But frequently, we don't know, because we sort of know our own products too well. And so we assume the best possible path through your application.

In reality, users are often confused about how to do something or why a control is a certain way, or they enter the data that's unexpected in a particular field. And they run into these corner cases that you didn't catch in testing, you didn't think about during development.

Typically, how that type of experience gets related, if it does at all, is through customer support or maybe somebody internally happens to run across it and then you're sort of left trying to guess how common is this? What steps actually led to the problem occurring? Has it been fixed since and so on?

[00:06:09] JM: You are the CTO of FullStory. Explain how FullStory works.

[00:06:14] MM: Sure. FullStory is a digital analytics platform, and that's kind of a new term for a lot of developers, because it's sort of a new space. And the way that I described it to a lot of developers is to say everyone is pretty familiar with the concept of observability now as a concept on the backend. FullStory and digital analytics platforms kind of extend the notion of observability on to the client side as well. Because what's going on in the backend is frequently just used as a proxy for understanding with the user experience is.

FullStory works, if you're a customer of ours, is you put a JavaScript snippet on your page. And the JavaScript that we run there does a number of things. But essentially, it is capturing the state of the environment as perfectly as possible inside of the user's browser or inside of the user's application and then sending that to us for analysis. So the structural content of what's happening with the experience on the page gets sent to us.

[00:07:22] JM: Okay. And how is that captured? Let's say I am a user I am navigating through an e-commerce page. Describe how the data is being collected and captured for a replay experience.

[00:07:38] MM: Right. It may be worthwhile to back up a tiny bit and talk about sort of the analytics approaches that a lot of people are more familiar with. Essentially, event-based analytics, where you're going in and tagging certain events that are happening. A user navigated to a certain page, a user clicked on a button, a user did a search, those sorts of things that are manual tagging.

The origin story of FullStory is that the founders were actually working on a completely different SaaS product, and we’re running into this problem of trying to understand what was working for users and what wasn't working for users. And they started going through this very familiar analytics, event-based analytics cycle, where you start to tag everything in there that you can think of, right? You come up with a hypothesis and then you put in your events. And then you collect a bunch of data and you wait and you look at it and you try to infer what's happening. And then frequently you say, “Well, that was inconclusive,” and you kind of go back to the beginning.

One of our founders said, “Hey, I think I can do something a little bit better than this.” If we were able to just automatically instrument everything that happened inside of their SaaS application, then we would be able to really understand with great fidelity what was happening. So that's what you did, and that was kind of the origin story.

So what does that look like kind of under the hood? What that means under the hood is when our JavaScript loads on the page, we’re taking a snapshot of the DOM at that point in time and sending it to our backend. And then we’re watching every mutation that's happening to the DOM and a couple other environmental signals. And that turns into essentially an event stream to our backend. Then on the backend, we have a process that is sort of re-creating the time series of that session and then pulling out analytics for it.

[00:09:38] JM: Capturing mutations from the frontend, describe how that works.

[00:09:42] MM: Yeah. Happily inside of browsers now, there is a great API for doing this. There is a DOM mutation observer. And so this is a kind of well-known API that you're able to hook into to get those state changes in the DOM.

[00:09:59] JM: It's funny. I talk to another company that is very similar to FullStory, LogRocket, and they have utilized the exact same browser API. It makes me wonder if there are any APIs that are being released into browsers soon that our other business opportunities. Do you follow the change log for changes to the browser that might present additional opportunities?

[00:10:25] MM: Yeah, we certainly do. I mean, we keep our eye on things that the W3C is doing and kind of always try to stay informed on that. There's a lot going on in the browser space as you know. So it's not an easy task to keep tabs on all of it, but there are a lot of interesting things that are happening. One of the ones that I was looking at probably about a week or so ago was some of the new work that's coming out around backend tracing IDs. So sort of taking the concept of open tracing and turning it into a browser API and a browser standard, which I think is fantastic.

[00:11:02] JM: FullStory has a number of different things that is going to give to the user, or to the developer I should say. So if I'm a developer and I am viewing a FullStory sessions, so that session has captured the changes to the browser that a user has gone to, like a user that is interfacing with an e-commerce website. If I am that developer and I'm looking at a FullStory session, what is my interaction with it and how am I detecting bugs that have emerged? Do I have to watch the entire session to understand what is going on?

[00:11:37] MM: No. That's generally not the most common case. Frequently, the way that you would find issues is by setting up searches for something in particular. So search is kind of at the heart of everything that we do. So you can define different searches for different criteria. So let's say you were interested in people who went on to your checkout page and started to fill out a form and then abandoned it and left without ever completing the purchase. You can set up a search for that, and then that defines a user population. And then you can look at analytics over time for that segment. And so you can start to dive in and start to say, “Okay. Well, where am I seeing increases? Where am I seeing decreases? Are there different facets of that population that may be accounting for some of the problems?” For example, are they all coming from a particular type of browser? Or are they all on a certain device type? Are they outside of the country where your service is operating? Things like that then lets you start to understand where you want to dive in and.

And then if you're trying to find a specific tactical issue, then yes, you can dive into the actual session replay, because part of the session replay beyond the DOM element is that we also have the JavaScript console in there and network activity so that you're able to understand at a very deep level what the issue was and how it got triggered and then where to investigate further.

[00:13:17] JM: And let's explore that idea of the session replay in more detail. The first step of the session replay is to capture the session replay. So on the user's browser, you're capturing all these change sets to what's going on in the browser. And then you have to shuttle all that information to the backend. Explain what are the engineering challenges in capturing that full lifecycle of a session replay.

[00:13:43] MM: There are many different ones. I would start with the issue of fidelity. So number one, in order to really understand what's going on for a user, you have to have a really high-fidelity and really high accuracy in what is being captured. So that’s piece number one. And that takes a ton of engineering work, and we’re really lucky to have just a bunch of browser experts who have spent a long time kind of honing the fidelity story for us and really understanding very deeply the nuances of browsers and how to accurately capture exactly what the experience is for the user. That’s piece number one.

Piece number two is doing that in an incredibly performant way. Obviously, you would never want any script on your page of any kind, whether it's analytics or anything else to interfere with the user experience. So we put a lot of time and care and testing to make sure that we are not interfering with the actual site experience. That's another great engineering challenge. How to do that across browsers? How to do that across sites of all sorts of various complexity? It's a very deep area.

The next one is privacy. Privacy is a huge component of what we do and what we really care about. And our approach to privacy starts with the philosophy of you never want any data to leave the user's browser that our customers didn't intend to leave the browser. So, nothing ever gets over the wire that our users don't want to. And there is a number of controls for that and making sure that those controls work and are flexible and robust is another area of challenge.

Efficiency over the wire, we talked about efficiency of the kind of capture side. But then as you turn these into event streams, the efficiency over the wire coming up to our services is another area that's really important. So how do you make sure you're sending the richest data you can in the most compact format that you can? And how do you do it in a way that's of course robust and tolerant to all the crazy things that can happen on a browser? You might lose network connection and you might be on a slow connection. How do you not miss any important data?

And then on the back end, of course, there's a host of all your typical large data problems on the backend. How do you ingest all this data? How do you reliably process it? How do you store it efficiently? How do you store it cost efficiently and make sure it's available and accessible to users whenever they need it?

[00:16:36] JM: The event streams, can you describe the data and the shape of the contents of an event stream in more detail?

[00:16:44] MM: Sure. The event stream is really just a representation of the mutations that are happening inside of the browser. So if you think of your browser and you think of the DOM that's loaded in there, it's a tree structure. And so you can describe the operations on that tree. Essentially, it's sort of like a dictionary of change this element in this way, or this element got added, this one got removed. Or here are some type of interaction that occurred.

Physically, the way that that actually gets manifest right now is actually in JSON. That's then compressed and sent over the wire to us. There is complexity with different browsers support for binary formats, and this is where the kind of art comes in of having to manage a large variety of browsers. Makes it a little more challenging in terms of the actual engineering representation of things.

[00:17:47] JM: And share a little bit more about the bandwidth that you can allocate towards that event stream. Because I can imagine, let's take the extreme example. Let’s say I’m playing an online game. That's going to have a ton of network connectivity if I'm putting an online game in the browser that has lots of different people interacting. It's going to have lots of changes to the screen that's going. And if you’re recording all of this data and all the change sets in the browser and turning that into session replay information and then shuttling that to the backend, that’s just a gigantic constant stream of data that I would expect you would have to throttle at some point. So how do you pick and choose what to record?

[00:18:29] MM: Sure. Yeah. A lot of that particular question comes down to a question of fit. If you're talking about an online game. It's probably not the actual sort of game canvas that is the most interesting thing to record. And you're right. I mean, it would be an enormous amount of bandwidth, because every pixel is changing all the time. And there's very low value in that. Instead, what you might want to do is really focus on just the control elements of the game. So it might be something like the game lobby or the game control that you would focus on instead and then actually elide all the parts that you didn't care about so that you would not be sending that data over the wire.

You’ have the structural information of the things that were elided, but not the actual content inside of it. Now, that said, gaming is also sort of not one of the sort of primary use cases of this technology, but it’s certainly a fair question.

[00:19:31] JM: On the backend, you’re collecting the data from lots of different users, from lots of different customers. So you have all these companies that are using FullStory. And then for each company, there is a number of different users that are having data collected from them so that session replay can occur. What is your system for partitioning those different datasets? Do you have some kind of Kafka buffer that's collecting influxes of all that data and then reading from it, or what’s your strategy for buffering and reading all that data?

[00:20:03] MM: The strategy that we have is all of our services are running in Google Cloud. And so we try and make the most of managed services where it makes sense. Cloud Bigtable is one of those services where we can use it as a buffer for the incoming data streams. And then further down the line of processing, there is a number of different databases that we use. But all of them have very strong partitioning so that you can have multitenant architectures in a way where that data never gets co-mingled, because that would certainly not be a good practice.

[00:20:44] JM: And why is Bigtable the primary source of storage?

[00:20:49] MM: Bigtable is a primary source of storage only on the frontend as a buffer before it gets processed. It was a good fit for the shape of the problem that we have, which was very high write frequency on the data. A lot of the data that's coming in is already partitioned by sort of keyvalue pairs. And so it makes it very easy for us to store and extract the buffered data that lives there. One of the nice things about Bigtable is that it's very, very battle hardened. Can perform at very high rates of reads and writes and has really high reliability.

[00:21:32] JM: Where does the data go after being thrown into Bigtable?

[00:21:36] MM: It works its way through a pipeline that we have that we frequently just call it the cook pipeline, because what happens is you take the raw event stream that's coming in and then process out the various pieces that you're most interested in. So some of that is extracting the bits that are more salient to a user experience and dropping parts that aren't.

And then another part is sort of higher-level analysis on that event stream. Essentially, synthesizing new events that are more semantically meaningful than just DOM events. So an example that I gave in this case to a lot of folks is if you're looking over my shoulder when I'm trying to buy a concert ticket and I click the buy button and I am clicking it repeatedly several times within a second or two, you can probably tell that I'm frustrated. That's a good indication that I'm frustrated. But the browser doesn't know that. In the event stream, you would just see that as repeated number of clicks over a certain period of time. You have to turn that into something semantically meaningful.

[SPONSOR MESSAGE]

[00:22:59] JM: Scaling a SQL cluster has historically been a difficult task. CockroachDB makes scaling your relational database much easier. CockroachDB is a distributed SQL database that makes it simple to build resilient, scalable applications quickly. CockroachDB is Postgres compatible, giving the same familiar SQL interface that database developers have used for years.

But unlike older databases, scaling with CockroachDB is handled within the database itself so you don't need to manage shards from your client application. Because the data is distributed, you won't lose data if a machine or data center goes down. CockroachDB is resilient and adaptable to any environment. You can host it on-prem, you can run it in a hybrid cloud and you can even deploy it across multiple clouds.

Some of the world's largest banks and massive online retailers and popular gaming platforms and developers from companies of all sizes trust CockroachDB with their most critical data. Sign up for a free 30-day trial and get a free T-shirt at cockroachlabs.com/sedaily.

Thanks to Cockroach Labs for being a sponsor, and nice work with CockroachDB.

[INTERVIEW CONTINUED]

[00:24:21] JM: These semantically meaningful patterns, so something like a rage click, or the health of an ecommerce website, or the number of times that I click on something and it doesn't display properly. These are things that you have to learn to extract from unstructured data. So what's the process for scanning this event stream of user behavior on the front end and then extracting value from it?

[00:24:49] MM: Right. A lot of it is we run FullStory on FullStory. No surprise. And so we have a lot of data going back years actually for our own site, and that is one of the richest test beds for us to understand user behavior. We can look at user behavior on our site, identify it frequently visually and then turning that into, "Hey, is this something that we can describe through a heuristic. Is this something that we can build a model for?" And then experimenting with the data in sort of straightforward ways and then rerunning sessions through these new analyzers and seeing what's your precision and recall curves look like.

[00:25:35] JM: Tell me more about that. So we need a precision and recall, I think those are terms that not everybody in the audience would understand. Could you define those terms and explain how they're useful?

[00:25:44] MM: Sure precision and recall are essentially, taken together, they define the accuracy of what you're doing. Most people are familiar with the concept of just whether the tool is accurate or not. So precision is really about how well you were categorizing something, and recall is the number of times that you're catching it essentially. There are strong mathematical definitions for these, but that's the jist. Taken together, they represent accuracy. Of course, you want to tune these things to be as precise as possible, but also having the highest recall possible. And that's always when it comes to data science, that's the ongoing challenge always.

[00:26:28] JM: Let's talk about the workflow for somebody who is using FullStory. If I am the developer and I find that there are lots of rage clicks occurring in a particular area of my application, what am I doing that information? How am I finding fixes and how does the deployment workflow, the solution and deployment, redeployment workflow go for somebody who's discovered an error after looking at their session captures?

[00:26:57] MM: Yeah. That's a really great question. The answer is frequently in most companies, it's bigger than just the developer. One of the key strengths of FullStory and technology like this is that it becomes sort of a lingua franca for all these different departments to be able to work together. If somebody is filing a support ticket that says, "Hey, I was trying to buy this concert ticket, and it wasn't working." That will go to a support agent who frequently if you connect something like Zendesk through an integration to FullStory, you're actually seeing the – you immediately have a link to the FullStory session where something happened. In fact, you have the moment in time in that session where it happened.

So a customer support agent may look at it and then decide, "This needs to go to engineering." They'll send it over to engineering. Engineering now doesn't have to guess based on just the written text from a user about the type of experience. They can actually see all the different things that led up to it. So the workflow for the developer, that point is frequently to dive in and see whether there is something on the screen, which makes it immediately obvious, or something in the event stream that triggered this issue. And because we also have the network activity of what's going on, you can correlate what happened in the frontend experience for that user to log so that you may have on the backend.

So you can tie those pieces together, and it tends to make solving those mysterious problems much faster and much more pleasant as a developer. So a good practice at that point, if you think that this is a widespread issue, is then to set up a search for the type of issue that you had. So if you know certain events that led up to that bug occurring for the user, then what you can do is set up a search that defines it. Find the actual population of people who have been affected by this and then set up an alert potentially on that where you can say, "Let me know when this trend either goes above or below a certain amount." And then you'll actually get notified when those things happen. So that's a nice way to say, "Okay. When I actually push out a fix to production, I can see this number go down, and then I know the issue is fixed for real."

[00:29:31] JM: Tell me a little bit more about the engineering stack on the FullStory side of things. So you mentioned the data gets collected into Bigtable, and then it gets run through a data processing pipeline to detect the kinds of common errors that might be presenting on the frontend. Tell a little bit more about the data infrastructure that's processing all that data.

[00:29:56] MM: Yeah. We are pretty big believers in simplicity. And, really, big drivers for simplicity is just doing things in a standard way. So our backend is all written in Go. We standardized on that language a long time ago. And so that makes everything very consistent and in very simple. Our frontend is only in Typescript as well. So that's pretty simple. Just having those two major languages dominate everything that we do makes life pretty easy.

As I said, we sort of rely on a number of different Google-managed services for different pieces. But, really, the heart of the system that we have on the backend is Solar. That's our main data store and our main index store for the session data that our customers are interacting with. As the data flows through different services, it's passing through a different analysis pipeline that's built of different microservices. Those microservices all run in Kubernetes. They run in GKE, and they're handing data off to one another, and eventually that data gets indexed inside of Solar.

[00:31:11] JM: What did you pick Solar?

[00:31:13] MM: Solar has a lot of really nice properties, and one of the key properties that it has is the extensibility and openness and the richness of these search platform. So we've highly customized different pieces of it that are optimized for the types of searches that we do. But we find using a document store for a lot of these events really makes sense for the types of use cases that our customers have.

[00:31:44] JM: Have there been any nuances to dealing with all the different frontend frameworks that a customer could be using? So if you need to be able to understand problems that occur in the rendering of an Angular component, or versus a Vue component, or versus a React component, the subtleties of each of those frameworks. Can you do that?

[00:32:08] MM: You can. We're actually working on something that is going to take that piece of it to the next level. Because a lot of times what you want to know is, is this a common problem that a lot of people have, or is this something that is unique to me? So we have some interesting efforts in that regard.

The nice thing about operating at the DOM level is that you tend to be agnostic to the particular framework that you're using. So, at a base level, because you're operating at that DOM, you don't necessarily care how the developer implemented the actual site. Now for a developer who's debugging, of course, there's a bunch of nuance. There can be certain bugs that are manifesting only in certain components on certain browsers, and that would certainly be information that you want to know.

And as I said, you want to know, is this a problem that is unique to you? Is it something that is coming from your code, or is it something that's coming from a framework layer? Because you're probably going to treat those things differently. So there is a lot of subtlety.

[00:33:18] JM: Are there any problems that you found working with maybe older technologies? Like JQuery?

[00:33:25] MM: That's a good question. None are immediately coming to mind. I mean, for us, a lot of the other issues or some of the limiting factors are actually older browsers. So you're going to have an old version of IE that is still very common on the web. And so in order for FullStory to operate reliably in that environment, it has to take into account the particular limitations of that browser.

So for example, earlier, we were talking about efficiency of the recording stream, and there are certain things that we would like to do across all browsers, but not all of them have the support that we would need to be even more efficient. So we're kind of – You have to wait for certain browser technologies that actually deprecate before you can move on.

[00:34:16] JM: Are there ever issues where the collection script or the DOM manipulation collection causes degraded performance for the end-user?

[00:34:28] MM: That is something that we guard against very, very heavily, and that we collect data on of course for ourselves to understand what the impact is to the end-user. That to us is one of the kind of cardinal sins that of course you always want to avoid. So we work really hard and we have a lot of test cases on the backend. We've built up a fairly large sort of compendium of different test cases that we run our recording scripts through to ensure that that's not the case.

[00:35:02] JM: When I started this podcast, I was under the impression that backend engineering was harder than frontend engineering. And one of things that has surprised me through the years is that the frontend has really proven to be quite a thick stack of technologies. Do you think this is a common misconception?

[00:35:23] MM: I think it's a common misconception by backend engineers, of which I am one, right? I usually describe myself as kind of a grumpy old server guy. And I think they're really just very different shapes of problems. On the backend, you have certain types of issues where you're worried about scale and you're worried about reliability and you're worried about availability of these different services. On the frontend, it's just a different set of constraints and a different set of problems. I think both of them are equally hard. But the thing that I have found particularly fascinating as I work at FullStory is to realize just how inaccurate a picture you can have of the user experience from looking at just backend server logs and not understanding the actual on the ground truth of users.

I had a funny case. This was going back several months, but I woke up one morning and was drinking my coffee and I had an alert from our web application firewall that said that we had blocked an IP because we saw a lot of suspicious activity from it. And specifically, this IP did something like 37 failed login attempts over the course of 60 seconds.

Of course, I just assumed that this was somebody who was breaking into our site and doing some dictionary attack or credential stuffing or something. And I felt really good that we had blocked this IP. And just to double check what happened, I took that IP and I put it into FullStory and just said, “Let me see if this person had ever actually had a session with us.” And I put it in, and what I actually saw was somebody legitimately trying to log in and running into a bug on our site. And then out of frustration, just clicking the login button over and over and over again, and we just kept sending the same XHR to our backend to log in.

Not only was this user running into a bug failing to login, but then we had actually locked this user out by blocking their IP. I mean, it was just a massive 180 for me to say, “Oh gosh! I went from a point of feeling super happy that we had blocked what was obviously a hacker into extreme empathy for the user who is running into problems on our site.”

[00:38:03] JM: Hey. I mean, sometimes just clicking that login button again and again and again, sometimes something different happens.

[00:38:09] MM: We've all done it, right?

[00:38:10] AD: Absolutely. The question I have to ask, and I discussed this some with the LogRocket guest who came on, is that there is this question about privacy around the collection of a full session replay. Being able to basically videotape what is going on in the browser for every given user. Has this been an issue that has been discussed throughout the FullStory organization?

[00:38:40] MM: Well, privacy is always top of mind for us. Number one, we're our own customers and we're our own users. And we wanted to do the right thing from a privacy perspective always. That's just the ethical thing to do. And so we always offer tools that allow people to dial in just the right amount of data that they need to collect from a given session. That's where it really is different. I know there's a little bit of sometimes even common misperception that what's actually happening is the pixels are being recorded on the screen, and that's not at all the case.

What's actually being captured are the structural information, and then any content that you also want to have. So we have a variety of different approaches that we take for this, where you can either dynamically or statically declare the different pieces that you don't want to leave the user's browser. Or you can flip it around the other way. We have an option called privacy by default. And privacy by default allows you to actually say only structural information will leave the user site. Not any content, unless you put it into an allow list for us.

[00:40:00] JM: Have there been any issues that have emerged in the GDPR space? What kinds of privacy protection rules have affected your engineering?

[00:40:10] MM: Well, certainly one gets called out most commonly is right to be forgotten laws and that aspect of GDPR. Of course, we offer APIs for our customers to be able to remove their customer's data if they get a right to be forgotten request.

The other part is around whether or not the user is consenting to have analytics recorded. And so there are a lot of sites that will ask you for that consent. We offer APIs at the recording level so that we don't actually begin any analytics capturing until the user gives consent. And even more interestingly, as part of that, you then as a site operator have an understanding of which users did actually grant you consents because you actually can have the specific session where it happened in a way that's easily accessible.

And so only when the user actually offers consent would we start capturing those – The analytics data for them. There's further subtlety in the API about the types of data that we record and when we can record it.

[SPONSOR MESSAGE]

[00:41:36] JM:

[INTERVIEW CONTINUED]

[00:41:35] JM: If you listen to this show, you are probably a software engineer or a data scientist. If you want to develop skills to build machine learning models, check out Springboard. Springboard is an online education program that gives you hands-on experience with creating and deploying machine learning models into production, and every student who goes through Springboard is paired with a mentor, a machine learning expert who gives that student one-on-one mentorship support over video.

The Springboard program offers a job guarantee in its career tracks, meaning that you do not have to pay until you secure a job in machine learning. If you're curious about transitioning into machine learning, go to softwareengineeringdaily.com/springboard. Listeners can get $500 in scholarship if they use the code AI Springboard. This scholarship is for 20 students who enroll by going to softwareengineeringdaily.com/springboard and enter the code AI springboard. It takes about 10 minutes to apply. It's free and it's awarded on a first-come first-served basis. If you're interested in transitioning into machine learning, go to softwareengineeringdaily.com/springboard.

Anyone who is interested and likes the idea of building and deploying machine learning models, deep learning models, you might like Springboard. Go to softwareengineeringdaily.com/springboard, and thank you to Springboard for being a sponsor.

[INTERVIEW CONTINUED]

[00:43:12] JM: The JavaScript the gets shipped to the user's browser, it’s minified often times. Does that make it hard to identify what is going wrong with the user's experience?

[00:43:28] MM: No, because we’re not actually looking at the JavaScript itself. The JavaScript is outside of the realm of the DOM. So we’re not looking at the actual code. What we’re looking at is the effect of the code on the structural information. So that makes our lives a little bit easier in some ways and is more relevant to what the user is actually experiencing. Happily, that is not kind of one of the major issues that we have to deal with.

[00:43:58] JM: You mentioned recording the network traffic. What kinds of problems can occur that you can detect from networking behavior?

[00:44:08] MM: Oh! There are a lot. Just to go back to the earlier example, I mean, certainly the way that I knew that a certain bug was manifesting on that login page was that I could see that we0 were sending an XHR. Even though the input fields haven't changed, we were repeatedly firing the same XHR over and over again. So those are the types of problems that you can get there.

Of course, you can look at return codes for those XHR calls and understanding that. You can see things along the lines of how long? What was the latency for a given call to get through? Browsers frequently throttle the number of open connections you can have on a site. So you can start to understand which order these calls were getting out and what the delays were between those calls.

Of course, also you have the potential to look at different headers that are coming back. Again, if you're trying to correlate the user experience that a particular user had in a browser and then correlate it with logs on the backend, you can do that through the network traffic analysis as well.

[00:45:16] JM: What are the goals for FullStory that you have not accomplished yet? What kinds of features of collecting data on the frontend can you not do yet?

[00:45:26] MM: It's less about the specific collection and more about the analysis of that collection. Everything that is impacting a user's experience is somehow represented in these streams. And so the job that we see ahead of us in FullStory is looking at higher and higher levels of understanding of those interactions. Why are certain people experiencing problems and where are people having better experiences and where are they running into trouble? And then trying to understand them at a higher categorical level.

So, for example, a product that we released several months ago is one called Conversions. And what conversions does is allow you to define a path through your site and then say, “Please analyze a variety of signals to help me understand, for people who were not successful in moving down this path, what were the main things that got in the way?” And that again may be things like they ran into dead click issues or rage click issues. They might have run into network latency. They may have run into different signals that you’re specifically telling us to watch for. And then we’re able to do a correlation analysis. And then that gives you a tool for actually diving in deeper and deeper and deeper.

So it's kind of the equivalent of looking at higher and higher levels of backend monitoring and analysis through APM type tools as supposed to printf debugging tools. So that's kind of the arc that I see full story moving up.

[00:47:13] JM: So you see a connecting backend problem manifestations with the frontend problem manifestations.

[00:47:19] MM: Yeah. Certain problems manifest that way, and certain ones are just frontend only. But most of the interesting ones are these kind of unknown issues that are in some interaction between your backend, the user's frontend and the data connection between them. It's really all three of those factors coming into play that add up to what the user is actually experiencing.

[00:47:46] JM: Any ideas on how to solve that? It all sounds like distributed tracing, but a distributed tracing where you would need to pass a trace ID from the frontend all the way through the backend.

[00:47:56] MM: Yeah. We have a lot of customers who actually do that part of it. And of course, we do that ourselves at FullStory, and that certainly makes it really easy when it's not a problem that's very obvious just from the frontend, and you have to understand what's going on on the backend. Yes, you can actually use those trace IDs to dig in and understand what was the flow once that request your data center and what might be compounding issues.

[00:48:23] JM: And how do people get that deployed? Did they set up Lightstep or some other distributed tracing system on the backend and they integrate it with FullStory, or is there some way that FullStory can manage trace IDs itself?

[00:48:36] MM: The trace IDs show up for us in a feature that we have called dev tools. So that's where you would see the actual network activity. And in order to make sure that dev tools can see it, you just have to tell us which header in particular you'd like to capture, because privacy extends to network as much as anything that is visible on screen. So by default, we are not capturing a lot of the network metadata. So you would tell us which header you didn't want to capture that representative your trace IDs. And then that would show up in the event stream for a given session.

[00:49:14] JM: Interesting. Tell me about a scalability issue that you've encountered in building FullStory.

[00:49:22] MM: Yeah. The types of scalability issues that we have tend to be related to the amount of data that we collect. We collect just vast amounts of interactions every day. Frequently, the biggest problem that we encounter is around the ingestion of that at the database layer. As I said, we've had to heavily modify Solar for it to meet the needs that we have. And so there is a lot of interesting engineering challenges in that piece.

Another piece is the actual analysis of the DOM elements and re-creating the experience that users had. That tends to be very memory-intensive, particularly on very complex sites. And so it's always kind of amazing to see the complexity that hides inside of the DOM on certain sites that you go to. It may not even be user visible, but there's a ton of complexity underneath. And so some of the scaling challenges are just around how do you efficiently process the DOM of literally hundreds of millions of sessions every day for super complex sites in a way that's efficient and not too memory intensive for the infrastructure you got?

[00:50:52] JM: Are there situations where you can end up capturing so much data for a given customer that it's a negative profitability?

[00:51:01] MM: We definitely work hard to stay out of that situation, and we've met a lot of optimizations where that tends not to be too much of an issue. But we do have to look at the complexity of a site and factor that into how our services are running and operating and what the cost basis is for it.

[00:51:22] JM: What cloud providers are you using?

[00:51:24] MM: Google Cloud.

[00:51:25] JM: And what caused you to opt for Google instead of AWS?

[00:51:29] MM: Well, I'll reveal a little bit of bias here and say that a lot of the original founding engineering team or ex-Google. And so GCP at that point certainly just felt a lot more like home, and the tooling was very familiar. And so that was sort of the natural fit. But in the many years that we've been on GCP, we just continue to love it. I mean, we think it offers a lot in terms of price performance and we're really happy with it as a provider.

[00:52:02] JM: From the business point of view, what has been the go-to-market like for FullStory?

[00:52:09] MM: Yeah. Go-to-market has changed over time certainly as the product has matured. Over the past three years, we've gone from kind of an S&P and midmarket customer-base predominantly into one that is ever and ever larger enterprises. And so the go-to-market changes a lot both in terms of the market segment that you're addressing and also just in terms of the use cases that people have in some of the organizational challenges.

Frequently, we find that there is an early adapter inside of a company that becomes your champion of a particular use case. And so we then work with those folks to understand the full value of FullStory. And then what frequently ends up happening is FullStory starts to spread very organically inside of the organization once everybody realizes this is not just a PM tool, or this is not just an engineering tool, or this is not just a UX or support tool, and everybody gets benefit out of looking at the analytics that are in there and able to share a common language for talking about what the user base is experiencing.

[00:53:25] JM: From all the session capture data, have you learned anything about what is in the average frontend stack or changes to frontend trends?

[00:53:36] MM: We just see a couple things at the aggregate level. But on the backend, we have to be extremely mindful and careful not to look at statistics across the entire user base. That's not our goal. We treat every customer individually. So we're not really an organization that is trying to find trends in different technology adaption across all of our customers. We tend to just focus on the insights so that we can deliver each customer individually.

[00:54:10] JM: Okay. Well, Michael, any last predictions for changes that will occur to the browser or to frontend tooling? Give me your most far-flung prediction.

[00:54:22] MM: The one that keeps me very, very excited is the notion that one day we will actually understand so much about user behavior that we will be able to test changes to our sites actually just in the CI pipeline by running something that's almost akin to kind of a Monte Carlo simulator so that you can test the effects of changes that you're making to the user experience before you even deploy them.

I think, hopefully, one day we'll look back and look at AB tests and think, "That was kind of a primitive approach." Imagine when you have to actually deploy a code in order to see the effect. It would be really nice to be able to move some of that analysis kind of left in your development pipeline into a CI or even development stage and predict what the effect of changes will be before pushing it live to your customers.

[00:55:18] JM: Michael, it's been a real pleasure talking to you, and I'm really proud of what you've done at FullStory.

[00:55:23] MM: Same here. I really enjoyed it, Jeff. Thanks for having me.

[END OF INTERVIEW]

[00:55:35] JM: Today's show is sponsored by StrongDM. Managing your remote team as they work from home can be difficult. You might be managing a gazillion SSH keys and database passwords and Kubernetes certs. So meet StrongDM. Manage and audit access to servers, databases and Kubernetes clusters no matter where your employees are. With StrongDM, you can easily extend your identity provider to manage infrastructure access. Automate onboarding, off-boarding and moving people within roles. These are annoying problems.

You can grant temporary access that automatically expires to your on-call teams. Admins get full auditability into anything anyone does. When they connect, what queries they run, what commands are typed? It's full visibility into everything. For SSH, and RDP, and Kubernetes, that means video replays. For databases, it's a single unified query log across all database management systems. StrongDM is used by companies like Hurst, Peloton, Betterment, Greenhouse and SoFi to manage access. It's more control and less hassle. StrongDM allows you to manage and audit remote access to infrastructure. Start your free 14-day trial today at strongdm.com/sedaily. That's strongdm.com/sedaily to start your free 14-day trial.

[END]