JARED: So, what was something about your Rails World experience that you'll never forget?

MATT: Well, I think it was when the DJ tried to sell me [censor beep].

JARED: [laughter]. I don't think we can say that on the podcast.

JARED: Welcome to Dead Code. My name is Jared Norman, and today we're doing a Rails World 2024 recap episode. I am joined by Matt Harvard, Sofia Besenski, Noah Silvera, and Chris Todorov. And we're just going to chat about, you know, what we learned, what was cool, and what's worth checking out when the talks drop for Rails World.

If you notice, my mic sounds a little bit different today. I am not recording in the usual setup. I am in the studio at my coworking space. So, you know, if you like it better, tweet at me that you like it better. If you hate it, tweet at me that you hate it. I could record here all the time if I wanted. With that, let's get into it.

All right, here we are. I have with me today our largest group that has ever been on Dead Code. We have four different guests today. Rather than go through all their intros, because there's just too many of you, I'm going to go through and ask each of you to describe yourself in a single word. Let's start with Sofia.

SOFIA: Interstellar.

JARED: Amazing. Chris.

CHRIS: I'm a sloth [laughter].

JARED: Noah.

NOAH: Excitable.

JARED: Accurate. And Matt.

MATT: Opinionated.

JARED: Oooh, all right. So, the reason I have brought Noah, Chris, Sofia, and Matt on the podcast today is the five of us were all at Rails World last week, or more like two weeks ago by the time this episode drops. So, we're just going to chat about what we saw there, what we enjoyed, what we liked, what we didn't like, and what we're looking forward to next year. So, we all arrived at the venue, which was in Toronto. It was the Evergreen Brick Works? Is that what it was called?

NOAH: Evergreen Brickworks, yeah.

JARED: Yeah. Very cool venue. Walking into there, what did all of you think?

SOFIA: I was wondering if I was at the right place. [laughter]

CHRIS: It was definitely a different environment from other tech conferences I have attended. It felt very open. And I think for the number of people that were there, it was quite cool. Unless I was lining up for food, I never really felt like I was fully surrounded by people.

JARED: Yeah, it was a good kind of mix of indoor and outdoor, which I think was really good, especially for...there is a fair bit of COVID going around and stuff to having, like, open air, like, the, you know, one stage being fully outside and everything, as well as just being able to enjoy the beautiful park that the venue was in. It was quite a cool spot.

NOAH: The Don Valley is pretty incredible. I learned there's over 70K of rogue mountain bike trails in there, too.

JARED: Ooh, that sounds like fun.

MATT: Yeah, so I had to walk through it to get to the venue every morning, and there were so many cyclists just ripping through it. Can confirm, bike trails.

SOFIA: It's in the middle of Toronto too, right? Like, this huge park.

NOAH: Pleasantly surprised. I thought it's not quite as integrated as Vancouver, but there's some large green space in Toronto that I feel like doesn't get talked about as often.

JARED: Yeah. Well, it's buried under snow part of the year [laughter]. So, let's chat about the keynote. So, we get in there; we get our badges, everything. There is, you know, usual conference fair, the booth set up from the sponsors, and then we all get into the main stage room there. And starting things off, of course, we have David Heinemeier Hansson, DHH, creator of Rails. What did he have to say for us? What did he have to tell us?

MATT: I felt like the theme from his keynote was, as far as looking in the future of what they want to do with Rails, was kind of, like, solidify what we have, really adhere to sort of the doctrine that's been written and make it more, I want to say, accessible for everybody. Like, a lot of the points he was hitting were like, oh, installing Ruby on a new computer isn't straightforward. Installing all these dependencies isn't straightforward. How do we make it better for everybody and keep making Rails what it is, a fun framework? That's kind of my take on it.

NOAH: I think you're dead on the money there. It's like, my understanding is Rails was designed to be a we deal with all your problems so you can focus on building stuff. And the idea of just let's move these things that you needed Redis for or needed a parse system for to you. So, you don't have to worry about that. Just develop. Put the queue in the database, put the cache in the database, throw your on-prem hardware and go.

CHRIS: Yeah. The move back to on prem that was probably one of the spicier takes, which I felt like there weren't enough. I almost agreed with everything DHH had to say, which I don't always tend to.

JARED: Testing wasn't dead this year [laughter].

CHRIS: No. And we can still use, you know, TypeScript and things like that. But anyways, I feel like that's one of the things that stood out with some of the tools, with Kamal and sort of that part of Rails that they're building out that. I mean, things like Heroku really easy to get going, but the cost of it and maybe the lack of innovation in the last few years on that side and the cost not going down has made it more compelling to start thinking about running things on your own managed hardware, talking about buying hardware and shipping it to data centers, which is definitely a more enthusiast kind of approach.

SOFIA: Yeah. I wasn't really sure about that kind of suggestion there. Kind of like ditching Heroku altogether and expecting everyone to just become, you know, sys admins and kind of do DIY to the fullest extent might've been a little bit enthusiastic in that kind of department. But yeah, I felt like it was very much kind of advocating for the more DIY kind of you don't need anything else but Rails kind of mentality, which I'm not sure is entirely realistic. But it's nice to think about when you're making MVPs and kind of doing smaller scale stuff.

JARED: Yeah, well, I think we, in particular, Noah, Chris, Sofia and I, we work on a lot of apps where some of these defaults that they're picking are not necessarily great defaults for us because we build e-commerce sites. And we're going to have some of those really, like, bursty, high volume traffic problems sort of out of the gate, which is going to make some of those choices, not necessarily all of them, but some of them a little iffier. And they would need to be rethought as a lot of the e-com businesses we work with would get to, you know, even medium scale.

But, Matt, I know you actually have experience with Kamal, unlike our team. What did you think of that side of things, Kamal 2.0 and the stuff that's coming?

MATT: I'm excited about it. So, right now Kamal uses traffic as a proxy, and then they'll be adding their own custom-made proxy. The idea behind it is that it will come with batteries included, you know, one of the Rails' doctrines there, in that, like, it can do SSL certificates from Let's Encrypt for you right away.

But we really like it because it does kind of give us that control of our hardware. Granted, we are using just EC2 instances on AWS. We could easily...we also have some Hetzner instances and some OVH as well, but they're just bare VPSs. And we can go from a bare VPS to a fully functional application running live in as long as it takes to do a Docker build. I mean, we really like it.

JARED: Yeah, no, it's something I'm curious to explore because I've never been happy with the Heroku path in terms of the price. You know, we've run things on AWS using various different offerings. They have...price-wise, that's better. But as you know, for a long time, we ran all of our stores on AWS OpsWorks, the last company we were at. AWS ended up deprecating that and creating a whole bunch of churn as businesses needed to move off of that.

So, having a more sort of reasonable default that gets people up and running without, you know, the costs of Heroku, especially since the Solidus stores we work on, are...they use some resources even out of the gate, let's say. They're a little chunkier than just a little Rails [inaudible 08:56] here. I'm optimistic. I'm going to keep an eye on that space for sure.

So, let's talk about a few of the other talks that people attended. There was one on scaling Postgres. There was one on Kamal proxy. We kind of touched on that already. There was a talk on Upgrading Rails 8 at Shopify. And then, there was Justin Searls' talk about The Empowered Programmer. What talk got you all most excited to jump in there?

CHRIS: Going beyond the single Postgres instance, I enjoyed the aspect of taking something, like, Rails already has some built in capabilities. And the speaker mentioned they were previously using Makara as a database proxy. But working through some of the challenges and how that didn't work for them, and deciding to build their own in house, something that I think most businesses would never attempt unless you're at the scale of Instacart, I guess.

But there were some interesting problems there with...and it kind of like...I used Makara on a previous project and had success with it. But it seems like it's not the perfect tool for all problems, specifically when you have multiple Read Replicas, and some of them can go down, which I've never had an experience. But when you're at the scale Instacart is, it's probably something that happens.

The main issue there was redistributing the traffic when instances are replaced and come back up for Read Replicas. And the existing solutions weren't adequate or required some proxy to be rebooted for the traffic to be perfectly redistributed. So, that was interesting. There was definitely a deep dive into what kind of algorithms you might need to consider and implement when you're building a database proxy, I guess.

JARED: He also showed off a good amount of sort of the different scaling pains that they saw over the course of getting their app to the scale that it's at, and classic answer, it depends, when it comes to, you know, optimization. He did a good job of sort of telling us what it depends on and why they did certain things at different points in scaling their app up.

Because, you know, naturally, their business is going to be different than the apps that we run and the apps that, you know, Matt runs. So, that journey is going to be different. And he was very, very clear about that, which I really liked. And, you know, he wasn't saying, like, these are the things you need to do to get your app to scale. He was saying, this is the problem we were having. Here's what we did and why.

CHRIS: Yeah, no, I think he did a really good job at covering sharding and, you know, when you might consider doing that. And definitely emphasized on think hard before you dive into any given solution that may be out there, just because a lot of time that might lead you into more trouble than actually help with the complexity you're introducing, of course.

JARED: So, Matt and Noah, I know you attended the upgrade handbook to Rails 8 talk. I was very curious about that one though I was in the other talk in that slot, and I'm going to check it out afterwards. But what was that talk about?

NOAH: That talk was about...so, Shopify has, like, 400 Rails apps, and they were struggling to get developers to upgrade them. So, they wanted to streamline the process. And I actually think this is a problem for almost all frameworks I've worked with is that upgrading is never as simple as NPM upgrade or a bundle upgrade. There's almost always extra steps that happen when it's a framework.

So, they basically built a step by step walk a dev through automatically open PRs with all the automated steps that they can do, process for upgrading Rails, and then a whole monitoring system along with that. So, that was essentially what Shopify did. And, of course, no one else is going to do that because no one else really has 400 Rails apps that they're managing [laughter]. But I really liked the process and I kind of wish...I was like, damn, I hope Shopify just builds this into Rails because why not for everyone?

JARED: Yeah, I'm curious. I mean, that sounds like a really cool idea, automating the upgrade process. But what was the takeaway from them? And what can we as, you know, we only maintain mere dozens of apps. And we maintain them across a variety of different organizations on different infrastructure. We might not be able to implement what they've got. What do we learn?

MATT: The big encouragement was to keep your apps up to date, because if you don't, you create technical debt. It could be technical debt, depending on how you want to look at it, later on down the road. So, that's really the point that they were trying to get across. I do wish that they would release that tool. That would be so cool. But, unfortunately, it's not for us mere mortals.

JARED: Well, it's been said on the podcast now that we want them to release it. So, maybe we have a listener at Shopify, I don't know. We'll see. We'll see. So, the other talk in that slot was Justin Searls' talk: The Empowered Programmer. So, first of all, I've got a podcast. Justin Searls has a podcast. That makes us co-workers. We all work at a big podcast. So, good to see one of my co-workers.

NOAH: Just to clarify, Jared, does this mean that all the other tech bro podcasts out there are also co-workers?

JARED: Yeah, Joe Rogan's another coworker of mine [laughter]. The [inaudible 14:52] girl she's also a co-worker now [laughter]. So, it's one big family. So, Justin gave a talk called The Empowered Programmer. And one of the sort of interesting things about that talk is that he says that it will be his last talk. He's retiring from giving conference talks. He's given quite a few really great ones over the years about testing and a lot of the work that he's done through Test Double. But he's done talks through a lot of work, and he's not wrong. I've given a bunch of them. A few of you have here as well. What did Justin have to say?

CHRIS: I think he was doing another one of his Rails is a one-person framework. And I think he might've even pitched it this way in his description. But this is the 2024 edition of that talk where he goes and builds an app himself and tries to demonstrate how much Rails gives you out of the box. And I thought for a newcomer, that would have been an excellent talk to attend just to kind of get a sense of what's there, like, a high-level overview.

For a more seasoned audience, it probably didn't contain a lot of new information, but it was still interesting to hear how he approached some of the problems, more from him being the technical team and his wife being the stakeholder. And a few of the things that I enjoyed was the decisions he made to not do something.

The one that specifically comes to mind was about storing particular data, and it was like, well, a lot of time, we try to be proactive and assume that some data will be needed and, as a result, create a maintenance burden forever. So, thinking of it a little bit more carefully about whether anyone needs certain information and only extracting what may be needed. In his example, I think he was storing some data in JSON and pulling out a few relevant pieces, and then throwing away the rest. So, I enjoyed that. It's something that, like, a pitfall that I've had before, in my experience.

SOFIA: Yeah, I remember that talk was...I found it really interesting. Like, I think I'm probably the most junior web programmer person in this room right now. So, like, from a less experienced perspective, I found it really interesting to see how, you know, casual he was about starting a new application, going from zero to production and actually taking people's orders and starting to make money for his wife, I guess, his household business, family business thing.

And kind of, yeah, for me, it was one of those talks that felt really...there's like tidbits of engineering advice and kind of, like, best practices in there. But it was also kind of a motivational inspirational kind of talk to kind of bridge the gap between your first kind of "Hello, World!" application that you write when you're just starting the Rails framework, and then actually seeing how maybe your instincts can kind of lead you down a path of tech debt and just frustration and how to kind of cut those corners strategically in order to kind of deliver something, get that MVP out, and stop, you know, shooting yourself in the foot, so to say.

JARED: Yeah, it was interesting to see his sort of technical approach. I think that was the thing that was most interesting is, you know, there's been a lot of pushback on using big JavaScript frameworks and just sort of overengineering the frontend in a lot of ways. And Rails is very

aligned with the more minimal approach, you know, whether it's just using UJS and normal HTML stuff or augmenting that with Hotwire, and Turbo, and stuff.

But he sort of didn't necessarily do everything the perfectly default normal Rails way. But he still kept things very well aligned with the framework and aligned with his goal of, you know, this not being a huge maintenance burden, which is something that having inherited a lot of Rails apps built by people who did not think like that, I wish more people thought like that. [laughter]

So, Day 1 finished with a fireside chat. It was Matz, the creator of Ruby, and DHH. And it was hosted by Tobi Lütke who is the founder of Shopify. Before that kicked off, Chris, here, won a suitcase [laughs], but he wasn't there for it. He decided he needed a nap and peaced out. Though he did eventually get his suitcase, don't worry. But the chat with Matz, and DHH, and Tobi touched on some really interesting topics. What were your favorites?

NOAH: I really liked hearing the creator of Ruby talking about how he just built the language for himself. Like [laughs], it was a selfish thing, and he built it with his opinions and his thoughts. And then, he was just very happy that so many other people liked what he did and [laughs] had similar thoughts. And it was originally just confined to Japan and the Japanese community, and then it spread out to the international community.

And, I don't know, Matz was just really cute. I think at one point he said, I think one of the best things Ruby's ever done is get the Japanese engineering community connected with the international engineering community. And he's just really happy that he helped facilitate that. He just seemed like a very, very sweet guy.

SOFIA: Every time I've seen Matz talk about anything, it is so heartwarming and feels like a little bit of a contrast between him and DHH, like these two kind of balancing. There's a balancing act there, and it's nice. Definitely a very nice counterbalance.

JARED: Yeah. There was a contrast for sure in the talk between the very, very sweet Japanese man on one side of the stage and sort of a much more forceful personality sitting right beside him. But they did share some ideas about what has made Ruby and Rails successful.

They were talking about how, in some ways, the way that they developed and didn't necessarily see a ton of traction initially, but then blew up and how they were able to sort of cement their good ideas and not back down from them, rather than becoming just a kitchen sink of other ideas or a design by committee situation. Both of them have, you know, single leaders with very strong visions for what they want, you know, Rails to be and what they want Ruby to be. And that has a really positive impact on both of those things sort of staying true to their vision.

NOAH: Yeah, definitely, I like the idea that having a singular unified vision can be really powerful for building some things particularly well. It does not work for building everything, but for some things like a programming language, I think it's really beneficial.

MATT: It’s pulling on that sort of origin thread with Matz and DHH. Like, Rails came from DHH trying to build web applications for the company he was working for, and he wasn't happy with how they were doing it, so he built something for himself. Similar to Matz being like, well, I want to build a programming language that I enjoy writing. So, I'm going to build that for myself. So, definitely some similarities, despite very different personalities at the of the day, as we know.

JARED: Yeah, yeah. So, moving into day 2, day 2 kicked off with Eileen giving a talk called The Myth of the Modular Monolith. And I think one of the things, before we talk about exactly what she was talking about, that's worth mentioning is that, like, a quarter of the conference was Shopify people. I think there was about a thousand people, you know, somewhere between 800 and 1,000 there, and maybe 200, at least, Shopify employees there. Can one of you tell me what was Eileen telling her co-workers on Friday morning?

CHRIS: I think, like, the TLDR is, you know, your problems aren't because, you know, you've made bad technical decisions. Your problems are human problems. You are personally responsible for everything wrong with your codebase.

JARED: Yeah, they're organizational problems.

CHRIS: Yes.

JARED: There are people problems to be solved. There's no silver bullets for architecting, you know, applications.

NOAH: And if you've got a hundred people contributing to a codebase every day, you can't enforce good technical design with rules and CI. It's like, that's something that's born out of culture and a desire to...

JARED: Only a hundred people? I think she was talking about when you've got thousands of people contributing to the codebase [laughter].

NOAH: True, yeah.

JARED: It's a lot easier when it's only a hundred.

[laughter]

JARED: Yeah, there was some spicy takes in that one. I think, you know, she talked about Packwerk, which is Shopify's tool for isolating packages within a Ruby application that you can control the interfaces to those packages, since everything is normally global in Ruby. And you can just reach into anywhere you want and access anything. It provides a layer that blocks calls across module or package boundaries, unless you've specifically allowed them, with the goal of making it so that you can isolate pieces.

I got the impression from her talk that, you know, there's nothing wrong necessarily with that tool, but it was sort of viewed as a silver bullet to the problem of, you know, scaling these really, really large applications like Shopify, and GitHub, and Gusto and other teams have. And that, it turns out just throwing a tool in there, not the solution [laughs].

CHRIS: Yeah, she was talking about how that tool doesn't...it kind of...if you're using it as intended and everyone's following the programming rules as well as the rules of the framework in force, it will work great. But there's definitely gaps in the tool that allow you to get around those restrictions. And it consistently introduced, you know, primitive obsession and, you know, other kinds of code smells to hack around those restrictions. So, yeah, that was interesting to see her kind of unpack some of that in a keynote in front of all of her co-workers and probably her boss.

MATT: Yeah. Tobi was there. Tobi was there [laughter].

NOAH: She was also talking about stuff like people just started delegating responsibility because of the modules. They're like, another team works on that. That's not my problem. That's not my tech debt. And it's just like, people are using as a way to almost shift blame was something I got from it. So, it's like, that's a risk of isolating code and specifically isolating modules to teams.

JARED: Yeah, yeah, you get into Conway's law, which is organizations which design systems are constrained to produce systems that are copies of the communication structures of those systems, and very, very much applies if you try to, you know, make your package boundaries your team boundaries as well.

NOAH: Modularizing people doesn't work as well as modularizing code.

JARED: [laughs]

NOAH: Sorry [laughs].

MATT: It felt like Packwerk, the technical pieces of Eileen's talk, were really just to illustrate all the cultural problems. That's how I interpreted that, like, Packwerk being a great example of everybody has their own module, but now they're saying that's not my problem because it's outside of my module.

CHRIS: Yeah. And, as we know, developers are pretty crafty and find ways to work around the boundaries the tools can sometimes try to enforce. And, as a result, you end up with probably worse code and spending a lot of time working around how the tool wants you to work, instead of actually thinking about improving the architecture or the APIs of the individual modules.

One thing that stood out to me, like, cultural, definitely organizational problem is the incentives aren't always there for developers to prioritize, like, improving architecture and maintenance.

And it's more aligned with shipping features fast because that's what gets you promoted or makes your work more visible. But, oftentimes, someone else has to maintain that code after you have been promoted as a result of delivering a large project. And, of course, other teams that have to interact with that part of the system as well are paying the price for it.

JARED: Absolutely. All right. So, day 2 is after Eileen's talk. I know some of us went, or some of you went, to Demystifying Some of the Magic Behind Rails with Ridhwana Khan. What was that talk about?

CHRIS: Ridhwana went into some of the meta programming in Rails and used a couple of really good examples like all the methods that are generated around ActiveRecord Associations, for example, to illustrate, well, sort of the magic of Rails, but also, how one can dive into the framework code, even though initially it might seem very overwhelming.

There's some structure to it, and it's relatively clear once you, you know, spend some time understanding these kinds of fundamentals that are probably reused among other sub-gems of Rails, not just ActiveRecord. So, it was, in a way, making a overwhelming task of looking at framework code more approachable.

SOFIA: More approachable, not, actually approachable, but [laughter] slightly more approachable. I found that talk to be, like, really interesting technically speaking. I think that's what a lot of the conference talks kind of lack for me, at least, is, like, I want to get in there. I want to leave this conference knowing more about the framework that I use every day. And yeah, so it definitely did make it more approachable or make the framework rather more approachable in the sense that I could go and actually look through the source code of Rails and start to understand how some of this magic happens, like when you boot up a Rails application.

And yeah, kind of seeing her process, I guess, about how she investigates, even from, you know, starting at the top level, starting from the API docs, and then not even touching the code until she has a good kind of higher-level UML diagram in her head about how all these things kind of touch each other.

And then, once she kind of has that idea, she starts to touch, you know, the source code and kind of, yeah, which makes sense because going straight into the source code for a dev like me is very intimidating. And it would take way too long to kind of get up to speed to make sense of anything. So, yeah, it was just another cool example of how, like, a more seasoned, more, you know, experienced engineer would kind of approach a complicated problem like that.

CHRIS: Yeah. And if I remember correctly, she's one of the main writers on the Ruby on Rails guides. And so, she was coming from that perspective of like, well, documenting these behaviors for new users or, you know, for existing users of the framework she needs to understand herself how the framework is built in order to better describe how actually parts of it work. So, that was a really cool perspective.

SOFIA: Yeah. I've learned a lot. Like, we have a person with technical writing background on, like, the Super Good team: Benjamin. And I've learned so much about how to kind of traverse codebases and think about, you know, very technical problems in a higher-level kind of context and kind of the value that you can gain from taking those kind of more abstract approaches to investigating technical issues. And, yes, this is just another example of kind of how adding the human understanding element to not just crunching through code is sometimes the best approach to kind of tackling the more complicated areas in a codebase.

JARED: That's Benjamin from Episode Two, by the way.

CHRIS: Shout out.

JARED: So, after that, you know, there were a bunch of really great talks. I think, you know, a few of us saw Andrea Fomera's talk on ActiveStorage. I thought that was really cool. She showed off some things that you could do with ActiveStorage to provide some sort of different backends to it, backed by various different services that kind of don't work the way ActiveStorage works, which I thought was really neat. I'm a big fan of Shrine.rb rather than ActiveStorage because I find it provides a little more flexibility. But it made me more curious about other things that, you know, you can do with ActiveStorage.

The next talk I want to talk about, though, is Robby Russell. Also, Robby Russell, who you may remember from Episode Three of Dead Code was giving a talk called Prepare to Tack - Steering Rails Apps out of Technical Debt. I don't know if talk's the right word. It was more of a guided meditation. But [laughter] what did you all like from that talk?

CHRIS: I think, for me, coming from a consulting background, some of the takeaways were very applicable to the challenges that we see in a lot of the codebases that we work on. Again, like a lot about being thoughtful when introducing features and just how they fit in, taking the time to make architectural changes, I guess, in order for those to fit in the application.

JARED: I believe his words were commit to never rewriting your fucking app.

[laughter]

MATT: Don't rewrite your application. It's never the solution. Yes, exactly. And I'm at a point sort of in my career where we tried that. We tried to rewrite our fucking app, and we probably shouldn't have. We've been working on it for five-plus years. And now we have two apps coexisting running against the same database. Do we have an end in sight? Not really. We've got a road map. Don't know when it's going to get completed. And we're just in this stage, and we have to maintain both of them. We don't have any options. So, that was a point that he made that really struck home for me because I'm like, yeah, I wish we didn't do that.

JARED: Yeah, he touched on some things that he touched on when he was on the podcast. He talked about sort of how you don't need permission to do your job well. If things need doing, bake them into the work that you have to do. If someone says, “How long is this going to take?” Don't say, “Oh, well, if I skip doing all of these things that definitely need to be done, then I can get it done faster.” If you can't get buy-in from your organization, this is the new cost of doing business. I think that works in some cases, it doesn't in others. You know, there's certain issues of trust around that where it's sort of, if that's the way you're forced to work, then you're already in a bit of a low trust environment, and that can cause issues.

I did like the perspective on the rewrites, the sort of broader idea that he had, that you should be making decisions as if you're going to be working on this app for the rest of your life. Somebody probably is going to be working on this app, and if you're not making decisions like you would in that mindset, then you're doing that person potentially a massive disservice. And I 100% agree with that.

MATT: Yeah, that makes sense.

NOAH: What's the oldest codebase we have inherited at Super Good? Like, what is the first command on our oldest codebase?

JARED: I don't know. There's definitely a bunch of Spree apps. So, they're going to be in the ballpark of, like, 15 to 20 years, somewhere in there, but I don't know exactly where.

NOAH: 15 to 20 years.

JARED: Yeah.

SOFIA: Spree, not Solidus.

JARED: Correct, yeah.

CHRIS: At least one started around 2012. But I'm sure we have some that are older than that.

JARED: Yeah, somewhere around there.

MATT: I think Our oldest is 2009.

JARED: Wow.

CHRIS: Did you get a t-shirt, Matt, with “I've been doing Rails since 0.8,” or something [laughter]?

MATT: So, I was very honest with myself when I ordered the t-shirts, and I looked back, and it was 1.0. Unfortunately, I wasn't in the 0.0 club, but it was 1.0. So, I did get some of those t-shirts, yes.

NOAH: Very silly, but I'm still proud of being in the Minecraft Alpha Club.

MATT: Oh yeah, yeah [laughter]. Nice.

JARED: So, after Robby's talk, we then got a great talk from Xavier Noria about the Rails boot process where he walked us through, you know, all the initializers and where Rail ties fit in and the inheritance tree around applications, and Rail ties, and engines, and all that. That was really cool. Definitely check that out if you're interested in, you know, what's happening with your Rails applications you're booting up.

There was talk about AI. There was a talk about testing integrations. But the last talk that I think we want to talk about here is Aaron Patterson's closing keynote. What was it that Aaron Patterson dug up for Rails here?

MATT: Webster Webb, the original.

JARED: So, what does Webster Webb tell me [laughter]?

MATT: What is it? Circa 2002?

JARED: I think so, yeah.

SOFIA: Myspace era.

MATT: Yeah, Myspace era. It was the first incarnation of what would become Ruby on Rails.

JARED: Yeah, pulled from digging into old Wayback Machine, Myspace accounts, and Tumblr accounts that belong to DHH.

NOAH: I can't tell, was that legitimate? Are those real accounts? I know some of it was photoshopped.

CHRIS: I want to know that, too, because I heard that it was a joke, and I definitely bought it. It was so good.

JARED: Yeah, I don't know if any of that was real. It is hard to tell. It was very well done though. But we got to see some real OG web tech in there including...I think the best part was the joke about RJS, where he showed off a technology that transformed Ruby code into JavaScript and was like, thank God that didn't make it into Rails 1, which was a joke because it did. That was a real thing. That was totally real.

NOAH: Oh my God.

JARED: That was a thing that existed in Rails.

MATT: So, I used RJS, and I liked it. Am I just, like, far out on this one? Like, I enjoyed it. I built some really cool things using RJS.

JARED: I think that it was a technology of a certain time, and it was pretty cool at the time, but I don't want it back.

MATT: Oh yeah, I agree with that. We're done with it.

JARED: [laughs]

MATT: But, at the time, I thought it was neat. Like, click a button, and something is happening. You don't need crazy JavaScript. Anyway. Okay [laughter].

NOAH: I loved transpilation when it was needed. I'm done with debugging transpiled code. You can't convince me to use a transpiler.

JARED: Technically, this wasn't transpilation because what was happening is the Ruby calls run and generate JS code. It's not like CoffeeScript or something. It's actually you have a limited set of things you can do, and the method calls actually generate JavaScript code.

SOFIA: It sounds like my favorite part about Deface [laughter].

JARED: This episode is already long. We can't talk about Deface.

CHRIS: Speaking of technologies [inaudible 40:51]

JARED: That's another episode. That's a piece of technology we'll have to discuss at some point. So, Aaron dove into all the work that he'd done or work that had been done, it wasn't necessarily all his, to make routing faster in Rails by using a recursive descent parser, a handwritten one to parse the...and build the routing tree or whatever. That was really cool. It was neat to see that used.

You know, we've got a new parser, Prism, in Ruby. We're seeing other parser work happening, making different parts of the framework faster. And that sort of connects my interview with Kevin where he was saying, you know, anything like, you know, technically splitting a string on spaces is parsing. There are probably places where we could use parsers for handling data that we consume. And it's cool to see people sort of making that a little more accessible, showing us that the parsers are kind of boring tech, even if you're building hand-rolled ones. That was awesome.

So, that closes out day two.

SOFIA: Does it? I think we're missing the last entry on that itinerary, the Shopify closing party.

JARED: Oh yes [laughter].

SOFIA: Featuring the robo dog.

JARED: Yes. The Boston Dynamics Terminator dog was present at the closing party at the Shopify office.

SOFIA: I think that was the most productive corporate party I had. Matt and I did some firmware upgrades on one of the pairing rooms.

JARED: Nice.

SOFIA: The door was requesting an update, and we followed it through to completion.

JARED: Yep. Well, no doubt Shopify will appreciate you doing the firmware updates on their door controllers.

SOFIA: Yeah, it really feels good to contribute to a party like that, you know?

NOAH: Does Shopify have Wi-Fi-connected doors?

MATT: I imagine they're Wi-Fi connected. Yeah.

NOAH: Are they a tech company [laughter]? That's terrifying.

MATT: I don't think it was the actual doors. It was like the tablet outside that would let you know if this room is booked or not.

SOFIA: Yeah, you could schedule...you could reserve rooms right off...

CHRIS: I really enjoyed the room with all the MIDI synthesizers. It had a bunch of setups, people jamming, and bookshelf with lots of bookshelf speakers, but also laptops with different software, one of which you can write Ruby to generate music, which I had not --

NOAH: Was is it Max for Live with Ableton?

CHRIS: I am not sure.

JARED: There's a thing out there that I've seen, I think, at RubyConf in Nashville or something. Someone did a talk on generating Ruby code or generating music with Ruby code. It's really cool.

CHRIS: Yeah, it looked like a fun, interactive room that, yeah, people were really enjoying.

SOFIA: To me, it kind of reminded me of a harsh noise concert that I've been to recently [laughter], you know, with all just such beautifully musically talented people really collaborating and definitely not playing music on top of each other. You know, it was great.

MATT: I think the bubble tea lounge did it for me. That was my highlight.

SOFIA: Yeah, that was good. Yep.

MATT: Or the crème brûlée station.

SOFIA: You had, like, five.

MATT: Yeah, definitely a couple too many.

SOFIA: [laughs]

MATT: But yeah, the pork belly on the roof was great.

JARED: Yeah, Shopify knows how to put on a party. So, I mean, overall, I think it was a fantastic conference. We got to see a bunch of what's coming up in Rails. There's a lot of excitement around making the core experience, the experience out of the box more accessible so that more people use Rails and can get into Rails, which I think is great. And, you know, there's other groups like Shopify and GitHub putting work into, you know, the experience at scale. So, it's great to see sort of both ends being covered and talked about and just a lot of excitement about them.

Next year, the conference will be back in Amsterdam, which I heard the venue was amazing last year. So, I haven't decided whether I'll be there, but certainly tempting. On that note, thanks, Chris, Noah, Sofia, Matt for coming on the podcast, chatting all about all the fun stuff that we did in Toronto.

SOFIA: Thanks for having me.

NOAH: Thanks for having me.

CHRIS: Yeah. Thanks for having us, Jared.

MATT: Yeah. Thanks.

JARED: Rails World was a ton of fun this year. It was great to see all the excitement around the platform. It was great to see everyone that, you know, I don't get to see, because we all work on Rails, but we work on it from around the world. There's a lot of familiar faces and people I'd met online but never in real life and people who I just hadn't seen in a while. So, it was a ton of fun. Hopefully, I'll get out to Amsterdam next year. We'll see. I got some other big trips planned next year, but maybe I could make it work. I need an excuse to hit up Europe.

This episode has been produced and edited by Mandy Moore.

Now go delete some…