EPISODE 26

[INTRODUCTION]

[0:00:00.6] JM: John Looney spent more than 10 years at Google. He started with infrastructure and was part of the team that migrated Google file system to Colossus, which was the successor to Google file system. Imagine migrating every piece of data on Google from one distributed file system to another. It's exhausting just to even think about.

In this episode, John sheds light on the engineering culture that has made Google so successful. He's got some very entertaining stories about cluster ops and site reliability engineering. Google's success in engineering is due to extremely high standards and a culture of intellectual honesty. With the volume of data in throughput that Google response to, one in a million events are likely to occur. There isn't room for sloppy practices or ignorance of edge cases.

John now works at Intercom where he is adjusting to the modern world of Google infrastructure for everyone. This conversation made me feel quite grateful to be an engineer in a time where everything is so much cheaper, so much easier, so much more performant than it was in the days when Google first had to build everything from scratch. It's interesting to hear John talk about working at Intercom now where it's just very easy for him to deploy something compared to when he worked at Google in the early days. He'd just had to build everything from scratch.

I had a great time talking to John and I hope he comes back on the show again in the future because it felt like we were just scratching the surface of his experience. It was one of those conversations that just flew by for me.

Software Engineering Daily is looking for sponsors for Q3. If your company has a product or a service or if you're hiring, Software Engineering Daily reaches 23,000 engineers listening daily and we would love to have your sponsorship if you've got something interesting to advertise to our listeners. Send me an email, jeff@softwareengineeringdaily.com if you're interested.

Now, let's get on with this episode with John Looney.

[SPONSOR MESSAGE]

[0:02:26.6] JM: Artificial intelligence is dramatically evolving the way that our world works, and to make AI easier and faster, we need new kinds of hardware and software, which is why Intel acquired Nervana Systems and its platform for deep learning.

Intel Nervana is hiring engineers to help develop a full stack for AI from chip design to software frameworks. Go to softwareengineeringdaily.com/intel to apply for an opening on the team. To learn more about the company, check out the interviews that I've conducted with its engineers. Those are also available at softwareengineeringdaily.com/intel. Come build the future with Intel Nervana. Go to softwareengineeringdaily.com/intel to apply now.

[INTERVIEW]

[0:03:17.5] JM: John Looney is a principal engineer at Intercom. John, welcome to Software Engineering Daily.

[0:03:21.8] JL: Thanks for having me, Jeff.

[0:03:23.2] JM: We connected after you had sent me some emails recommending some topics to cover and you and I talked in the Software Engineering Daily Slack channel, and I found out that you had been at Google for more than a decade, from 2005 to 2016. What I'd I like to discuss first is the infrastructure in those early days, because there were so many interesting papers that came out of Google and lots of interesting stories. What was Google like back in 2005?

[0:03:55.4] JL: It was an utterly different planet. I'd come from a really tiny company and straight into Google with the follow on Silicon Valley culture was something I was not prepared for. That's to put it mildly. The Irish tech scene is incredibly tightknit. Everyone knows everybody. Usually, your reputation precedes you and does very little job interviews back then. Everyone knew each other, while Google was super professional. The interview process was terribly

grueling and the sense of culture at that time was just all around. Everyone just knew how to behave. Everyone knew what actions to take in any given scenario. It was a hell of a transition.

[0:04:41.6] JM: From the outside looking in, Google looked really stable. It looked pretty bulletproof if I remember by our standards back then, certainly compared to other sites. Maybe I'm remembering this incorrectly. What were the resiliency best practices for the Google infrastructure in 2005?

[0:05:02.1] JL: Google was a simpler place. I supposed at the time, this is even Gmail was only in beta. The only two products that had was search and that, and search is technically stateless. It's very easy to keep it up and running compared to more maybe dynamic applications. Back then you do the giant web index, crunch the numbers, or crunch over the Internet, push out an index every few hours or every few days. It wasn't that complicated. It was certainly something that was a challenge to keep up at 5/9s, but it wasn't impossible.

Ads was something similar, a lot of off-line processing and then you push your stateful ads and we're done. That side of thing seemed incredibly professional even then. There was a lot of low-level infrastructures. Today, with the cloud computing, people are used to building systems from dozens of small distributed systems components. Even in Google 10, 12 years ago, those didn't exist. In the early days, it was search building infrastructure for themselves, ads building infrastructure for themselves and occasionally people discovering that components were reusable.

[0:06:15.6] JM: This is almost hard to imagine. I guess back then, Google was mostly search and the ads they got displayed along with the 10 blue links and that was it. It wasn't this multifaceted email and consumer product and YouTube and all these other things, it was just search and then search advertising.

[0:06:40.7] JL: Yeah, you could even think of where has search come in that time. Back then, if your webpage is important, maybe it got indexed every day or two. If it wasn't, maybe you might go two or three months without seeing the Google bot crawling you, and people thought that was fine. The hardware that we're using back then, it's still pretty common for machines to have two gigs of RAM, so it wasn't like you needed a complicated cluster architecture to orchestrate

different size jobs in different machines because, no, you just make it fit in one and a half gigs of RAM and there you go.

[0:07:17.4] JM: For people who don't know, the way that Google used to talk about infrastructure publicly was that Google would make an engineering breakthrough and then they would stabilize it internally and then maybe they would talk about a little bit and then they would write a paper about it. I don't remember when they started doing this. I guess this was like 2007. When the first paper come out? Was it map reduce?

[0:07:44.2] JL: I think it was a conference around 2006 and we had the bright idea of getting all of the white papers we could find, printing about and handing them out for people to read just at a recruiting table at a conference. There were probably any four or five papers even back then. Yeah, map reduce designing planet scale data centers and even Google file system I think was pretty new back them.

[0:08:09.2] JM: I remember reading these in a distributed systems class I took in college, and I guess I didn't understand at the time that that was kind of — Actually, I don't even know if that was a new thing. Did Microsoft do that stuff? Who else published those kinds of corporate papers, that sort of corporate —

[0:08:28.8] JL: I'm thinking HP and IBM certainly did. HP had some really great ones that I would've read in college in the 90s, in the early 2000s. It certainly become a badge unless your organization is publishing, you won't get the engineers who will have real pride in their work, they won't come work with you. You could almost say that back in the day, things like Bell Labs, there were specific research type areas or research type companies that have big research departments and they published and maybe the corporate side didn't. HP would certainly write great papers on how you build enterprise solutions for backups for instance, but they weren't famous for who designed printers, publishing, that sort of content. While now it seems everyone has to publish about their speciality or they don't get attention from the press.

[0:09:26.2] JM: When you were joining Google in those earlier days, did you have the ability to read those papers, because it takes some — In my experience, it takes some practice to be able to — I still can barely read some of these academic papers. I can read the abstract and

then maybe I read the first or the second page and then after that I'm not great at reading those papers. I think it takes a certain amount of effort to be able to parse through the papers and make the most of it.

[0:09:54.9] JL: Definitely. I helped out on the Lisa conference a few years in a row and they get a good view academic papers on some very very interesting technical topics, but you read 20 of those over two or three days and your head is melted.

Yeah, I think in the early days, I certainly wouldn't of been someone who went looking for those kind of papers even if they did exist. I was a belt and braces old-school sysadmin, so when I joined SRE team, the concept of SRE was rather new to me and it was like, “Hold on a minute. I'm going to be on a team with software engineers?” and it wasn't just software engineers, these were badass software engineers who had way better distributed systems and algorithms knowledge than I did. I was just trying to help them initially, get up to speed with operations and understand that just because it looks good on paper and it pass unit tests, it doesn’t mean it’s good to deploy to the world. They weren’t a lot of papers back then on that sort of synergy.

[0:10:51.1] JM: Back then, the role that is now referred to as SRE, the site reliability engineer, was called cluster ops. What was that role like?

[0:11:04.9] JL: SRE came from the production team which were all software engineers in the early days. Yeah, they enjoyed operations, but was it was very much from that side of the house. Cluster apps was then spin up to look after the maybe less technical side of things.

[0:11:20.4] JM: So this is different.

[0:11:21.7] JL: It was initially a separate team that looked after — you might consider them unimportant things like the Google file system, like Borg, like the production image and any of the low-level . It was more of an outgrowth of the data center operations teams.

Now — Yeah, I think there was a debate even, did cluster apps need to code? Et cetera. My first task on the team was, “Well, here's the list of 10 or 20,000 broken machines. See if you can fix

them and get them back serving," and it was quite intimidating and quickly learned I need learn Python and I need to be able to script my way out of this or I'm doomed.

Then I came face-to-face with these professional software engineers who said, "No, you're not scripting workarounds. No hacks. If you find a bug like 4,000 machines are broken because if a kernel bug is filling up their hard disc, go fix the kernel bug. Get that patch deployed to the — Mainstreamed into the Google version of the kernel and deploy that everywhere. Don't do workarounds. Solve the root cause." That was one of those big early Google culture things that was very hard for me.

[0:12:29.5] JM: That's incredible. Did that kind of change when you make a fix to the Linux kernel in the Google version of the Linux kernel? Was there a standardized way of pushing that back out to the Linux community? What was that feedback loop like between Google and Linux?

[0:12:44.9] JL: In the early days, like any company when they're growing, you start off with a few engineers as you can handle. If you only have two or three full-time kernel engineers and they're getting told, "Oh, you got a new hardware platform to qualify. Make sure the kernel works on that." Other people to bringing in giant bugs like — I don't know. Map reduce on this architecture causes it to crash. They didn't have a lot of time to upstream books.

In later times, Google realized how important it was because I think the 2-4 to 2-6 transition would have been a real pain. We would have hundreds if not thousands of small little patches made by everyone from me with one line logging fixes to rather more competitive memory architecture changes that would have been put into these kernels and they went, "Okay, we're going to have to allocate real resources to upstreaming things quickly."

Today, I think you see huge numbers of Google patches to the mainstreamed Linux kernel. I think it's just a natural thing maybe on the scale of companies and maybe where their goals are focused. If they are large and they can afford the people, absolutely. If they're focused on maintaining a really great relationship with the open-source community, well, yeah. If your thought priorities and other things, like keeping the website up and running, keeping your ads generating revenue, then, yeah, upstreaming to the Linux kernel is not going to make the top 20 or top 100 priorities.

[0:14:10.5] JM: I think the incentives are aligned there because, certainly, Google would not want to have their version of Linux significantly forked from the open-source version, because then if the open-source version makes some patch fix where there's a merge conflict between the Google version and that patch, then Google would have to figure out how to resolve that merge conflict internally and then that's just a can of worms. Yeah, it makes sense that this is just something that would resolve with scale overtime.

You mentioned in an email thread that we had that you were part of this project to migrate Google from GFS, which is the Google file system. This is distributed file system that breaks files into chunks and then it replicates those chunks in —I guess it's just sharding and replication for files and then you migrate it — You help migrate that Google file system to Colossus, which was the successor to the Google file system. This is a massive project. You had to migrate all of Google. That's all that Google is, is files.

[0:15:23.4] JL: Yeah, it was literally a colossal projects, if you don't mind the pun. There's actually a chapter — Its covered slightly in the SRE book, but we couldn't tell the whole story because that would be a book in itself. That was a fascinating, fascinating insight into Google's culture because we knew it was going to be an enormous task. It's one of the ways that I like how Google's round services, like the Google file system is run by a storage SRE team that look after.

A lot of these service teams, their SRE is looking to make a name for themselves. We're chatting one day to an X-GFS developer who had moved on this Colossus project. It was just a research file system. The best way to describe it was it wasn't very good. It didn't have renames. You couldn't append files. You didn't have directory. It was kind of a pretty weak sounding file system, but it was really efficient, really low latency and there was a lot of scaling things in it that we liked. It was considered research. No one was going to use it seriously. I was scratching my head one day and went, "Hold on a minute, YouTube would really like to talk to guys because they didn't tend to change files. They don't really need directories. They've got all their files in a database anyway." I don't think they append videos.

Yeah, if you can make it lower latency and have the storage requirements or whatever their policy at the time, there you go, there's your first customer. We were able to boot kind of a — Almost like a startup SRE team, like let's be Colossus SRE. Give it a go. See how works. If we can get customers, if we can get people inside Google who want to use service and we provide something good, where we stand over the software. We make it go. We provide the tooling to give you quota isolation, 24/7 support, all of these things that want out of cloud product, then yeah, they'll come to us. They'll say we need help with this, and you build a team around it I suppose. It's a real startup mentality inside of SRE which is really really cool.

[SPONSOR MESSAGE]

[0:17:27.6] JM: Blockchains are a fundamental breakthrough in computer science and Consensys Academy is the place to learn about block chains. The Consensys Academy developer program is a free, highly selective, and carefully designed 10-week online curriculum where you will immerse yourself in blockchain development and earn a Consensys blockchain certification.

By completing the program, you will be eligible for immediate hire by Consensys; a leader in the blockchain space focused on the Ethereum platform. If you want to learn about blockchains and become a developer for blockchain technology, check out the Consensys Academy by going to softwareengineeringdaily.com/blockchain. The graduation ceremony is a spectacular all-expenses-paid trip to Dubai where you will meet block chain developers working on real-world solutions.

Dubai has announced ambitious plans to become the first city to run on blockchains by 2020. If you like the idea of immersing yourself in blockchain technology, graduating, and going to Dubai, check out softwareengineeringdaily.com/blockchain. Build your new career on the block chain with the Consensys Academy developer program. Applications are open from now until July 1st, so you want to apply soon.

For more details and to apply now, go to softwareengineeringdaily.com/blockchain. To learn more about Consensys and Ethereum, please visit Consensys.net, and sign up for the Consensys weekly newsletter for everything you need to know about the block chain space.

Thanks to Consensys for being a sponsor of Software Engineering Daily.

[INTERVIEW CONTINUED]

[0:19:28.5] JM: YouTube was the MVP of the Colossus migration.

[0:19:33.3] JL: It was.

[0:19:35.0] JM: I guess was YouTube even on GFS at that point or were they just on their own infrastructure?

[0:19:41.0] JL: They were on GFS in the in the very early. GFS was an interesting product in itself because it wasn't designed as a general-purpose cluster file system. It was purely for storing the indexes for web search. Of course, 10 years ago, Google didn't have 100 people to throw at GFS. It just had two or three really great developers who were making something for the indexing people. To make that into a general-purpose file system that YouTube and eventually thousands of others could use concurrently without stepping on each of his toes, that was a phenomenal achievement.

One of the big transitions that ever — I left Google a few months ago to join a small Irish company called Intercom, and I'm still getting used to the differences. One thing that I miss from Google is that all software is written incredibly professionally, so everything is intended to be shared. Big table Colossus spanner and the likes. When someone is starting a new piece of infrastructure, of course they're going to have quotas. Everything has to be — You'll have memory quota, CPU quota, RPCs per second quota. All of these things have to be built into the software from day one or no one will use it, no one will touch it.

I'm now playing with things like elastic search and MongoDB and I'm scratching my head going, "How do I set the — or PCs per second based on different users or based on different IP addresses?" and people are looking at me going, "What?"

Things that I would've accepted, no SRE in Google would accept a service if it couldn't protect itself from overload and denial of service attacks, and yet here's so much software that the rest the world is using now in the open-source community that is ridiculously brittle.

[0:21:23.9] JM: But its web scale.

[0:21:25.8] JL: People have different ideas for the what web scale is. Web scale in ideal world, as long as you don't hit it too hard and as long as you provision it properly in advanced. While in Google, it very much gotten used to software that you could beat with hammer, or a thousand people could use concurrently.

[0:21:42.7] JM: Yeah, or at least because of the quota, you know what size hammer is going to break this.

[0:21:50.2] JL: Exactly, and we would even do things like overselling capacity. On the Colossus team, we would know for instance that, yes, your average user probably going to use 60% to the quota. Great. We'll sail 140% of the cluster capacity, and we know that's fine.

Also, someone is going to go kickoff at 5,000 shard map reduce and they're writing really really heavily, but they've only paid for 200 spindles of disc capacity or disc throughput. If there's more there, they get it. If there's not more there, they'll get rate limited and the people who are doing low latency and say, "I want to use this for low latency. I'm only doing 100 RPCs a second but I want every one of them to succeed in 20 milliseconds," that's possible. Most, unfortunately, of the open-source software, people have not considered that to be important at all.

[0:22:42.9] JM: Google never really had a move fast and break things philosophy. It's sounds like it was always set quotas and meet your guarantees.

[0:22:53.0] JL: I would say that clear communication with hard commitments was definitely encouraged. When you said — Yeah, this is a cluster files system. Actually, GFS in early days, as I said, it was written for the indexing people, so latency wasn't considered important. It didn't have isolation. The early versions didn't even have file permission. When people said, "DO you have an SLA for this?" I'd scratch my head and I think at point I was saying, "Yeah, your reads

will come in around 300 milliseconds, and we'll give you a 90% uptime," and people were horrified. It's like, "That's what does right now. Would you like something better?" "Yes." "Okay, let's write a case for this." Go back to the developers and say, "Right, GFS version 2, that's what we get, but maybe GFS version 3 could make changes where we could get it up to 99% reliable and maybe latency down to 100 milliseconds."

Over the years, that kind of clear communication for development teams would come to various storage SRE teams and say, "But I have this use case and it's worth $10 million to me if you can make these changes and maybe get your legacy down to 10 milliseconds for reads."

Overtime, that software has become incredibly good, and you can see if you play around with cloud engine now, I think the sub-millisecond reads from the cluster file system or maybe even far better than that these days. I haven't actually gone benchmarking. That's because of that re-clear feedback loop between SRE almost acting like the product managers for infrastructure and the development teams finally got a reason to implement various features, because certainly most of the early development teams, it wouldn't have occurred to concentrate on things like latency. What they were adding features, like snapchatting and the likes.

SRE having talked to teams saying, "We can't serve in GFS. We have to use RAM, and that's really really expensive. But if you could get the latency down below — I don't know. Hundred millisecond reads guaranteed, well then we could serve things that can tolerate high latency from GFS and save a lot of RAM." That kind of concise clear communication channels with commitments articulated through SLAs allowed that to happen.

[0:25:02.1] JM: When you were doing the migration, when you were migrating — I guess first, you migrated YouTube from GFS to Colossus, and then you probably had some learnings there and then gradually you had to migrate all of Google as you proved the migration path worked. What were you learning throughout that process and what is it take to migrate everything on Google in 2005 to 2006, whenever that was?

[0:25:32.4] JL: A friend did an amazing kind of summary of where we were at one point, and it was rather beautiful. We were describing it as a moonshot, something enormous. We're not sure we can do it. We're not sure when we can do it, but were going to give it a go. Halfway

through, we did a state and nation, “Here’s the good news. We’ve moved so many petabytes from GFS or so many exabytes from GFS to Colossus. Here’s the bad news, Google has — Sorry. We're halfway toward the moon. Well, no. We’re halfway toward the moon was a year ago when we started. Now, we've added three more exabytes or something crazy. We’re only one- third of the way, so we need to accelerate progress.”

When you mentioned earlier about move fast and break things, one of the low points of the project were as a result of an unfortunate confluence of bad hardware, bad batteries, bad look, bad tooling, a very aggressive schedule and two power cut simultaneously. It took out a whole data center. Storage node is complete the off-line for about three hours. Thankfully — I’m pretty sure most services had backups at that time too with their data centers, but it was still very very scary, and it was three long days and three long nights before we had everything put back together.

The team was utterly demoralized and dejected on both sides of the Atlantic, and of the best managers that I ever worked with, he slapped me on the shoulder, he laughed and he said, “John, you just learned more in the last three days about your new service than you had in the previous three months. I think that's worth celebrating.”

I thought about it, we bought a few —Went downstairs, bought three bottles of the cheapest, nastiest sparkling wine, brought the team into a conference point, pointing at a whiteboard and said, “Okay, start writing down everything that we learned on the whiteboard,” and we toasted everything we’ve wrote down. It was pretty incredible to turn around the team like.

While move fast and break things is not a motto, it’s certainly used from time to time and the culture is there, that yes, sometimes it goes horribly wrong. As long as you learn from it, that’s school.

[0:27:46.7] JM: Totally. I remember having those kinds of —I haven't had anything like that crazy happened to me at a job, but certainly I remember in school, those nights where I'll be staying up late working with fellow students just on some — I mean I think this is why a lot of us fall in love with computer science or fall in love with software engineering is those late night

moments where you're totally frustrated. You're almost out of energies. There are kinds of moments where you have to laugh or cry, and so you choose to laugh about it.

Give me another crazy story, because that's just a crazy story, and I would to hear — Give me some other crazy tidbits.

[0:28:34.9] JL: One of the things that I learned on those storage seems in the cluster. We used to look after [inaudible 0:28:42.7] on that as well, was some problem — Some one in a million problems. When you've got near million machines, they're going to happen pretty frequently. One of the more demented ones was when one day one of our larger big tables just disappeared and everything fell over. Everyone freaked out. It's like, "Where did this go? What happened?"

Someone checked the logs, and there we go, a delete call was made and it deleted the whole big table. This would've been over tens of thousands of machines and enormous, enormous database. They froze everything, tried to work at who sent the big table RPC, but when they looked at it they couldn't find any mention of the RPC coming in. The call was made but there was — Definitely, no one actually kicked off the request. Someone opened up a debugger, the master process was still running and they examined the stack, they examined the heap looking through various bits and pieces. What they found was that the delete constructor on the big table object was 32 bytes away from the — Our create new row constructor.

What they think happened was the cosmic ray flipped a single bit on a single register causing the offset to the — Instead of calling the ad constructor, it deleted the big table. That was seriously scary because everyone turned around going, "How do we protect against cosmic rays flipping individual bits on a register when you're not ready for it?"

Of course, these cosmic rays or minor bit flips, sometimes they're permanent. There was another machine where for some reason it was just always running very slow and we logged in, had a look, and in the logs it was regenerating checksums for Google file system chunks, I think, repeatedly. We're trying to work out what happened. Restarted the job and it went away. People just assumed, "Okay, one-off clip. Whatever." A few days later it came back, and so someone had a look. I don't know how — Some people are just natural troubleshooters, but he

noticed that the problem only happened when the chunks server process was pinned to core one. On core, anyway, he tried to work out, “What’s the checksum code,” and got it down to five or six assemblies, maybe 12 assembly language instructions that implement the checksum code.

Yeah, when it ran in core zero two or three, it was fine. On core one, though the checksum would be incorrect, and it turns out that, yeah, the load effective address 32 instruction had somehow been damaged on it was flipping an extra bit in the high bit. The chip was sent back to the manufacture and it’s like, “What the heck?” They run it through their standard testing process and went, “Yeah, this would have failed our testing when it left the factory. It’s definitely broken now. Could have been a cosmic ray doing permanent damage to one of the transistors on the chip. These things happen.” How do you build in software to cater for those kind of one in a million events?

I used to laugh, because Twitter — Was it Twitter invented the concept of the Chaos Monkey?

[0:31:46.9] JM: No, that was Netflix.

[0:31:47.9] JL: Netflix. I remember laughing at that thinking, “We had enough machines and enough software and enough people. We don’t need Chaos Monkeys.” They would just spontaneously evolve like [inaudible 0:31:57.6] brains in our cluster infrastructure.

[0:31:59.5] JM: I think there’s another thing we discussed over email is that the statistical tales in Google infrastructure are almost guaranteed to happen. A one in a million event is going to happen daily. How does that affect your day-to-day existence as an SRE, because eventually you were an SRE at Google and if you’re talking about reliability, you got to be reliable in the face of the one in a million events, like cosmic rays.

[0:32:27.2] JL: Yeah. I suppose you start off getting very paranoid or you’re scared about everything, and after a while maybe your adrenal gland gives up and you realize, “Yup, anything could go wrong and anything will.” Rather trying to prepare for anything, let's just train for everything.

What I mean by that is just get good at incident management, get good at triage and get good at working as a team and get good at pulling in people as needed, because you'd never know what skills and what experience you're going to need in advance.

It kind of feels terrible now, but yeah, I used to be on outserving team for a while and having the pager for a service that's thousands of dollars a second is really frightening when you think about it, but you still have to just get over it and say, "It'll be fine. If something breaks, I'll be able to fix half of it in a few minutes. Anything else, will pull in the team, and every problem we've ever come across we solved it eventually."

Yeah, they spent a lot —Google certainly invested an awful lot in I would call soft skills and teambuilding training and it really paid off.

[0:33:35.8] JM: At a certain point, you got into SRE training and the management of SRE's. What are the reproducible strategies for training a site reliability engineer and managing cyber liability engineers and making that's a reproducible cultural thing that's built into the company rather than tied to any specific engineer?

[0:34:01.7] JL: There's definitely — Any large organization is going to have a very wide spectrum of management styles, management skills. You'll have the by the book people who they — I wouldn't say micromanage, but they certainly like to plan out things and know exactly where everyone is, and then you'll have the coaches where projects aren't terribly plan but everyone gets mentoring and everyone gets loved. In their own different ways, teams can succeed like that.

The common trends around SRE and SRE managers is that the managers themselves tend to be very technical. Google always had this idea that you get to senior engineer first and then maybe most senior 10% of engineering before you can move in as a manager. This helps avoid problems where engineers doubt their manager's competency over technical matters and as well as having a baseline respect means that the manager has credibility to rule on decisions.

When you're having an urgent incident happening, having people fall in naturally to kind of a military style command structure for a while is really really helpful. Then a manager that can flex

their style once that big incident is over and then flip into the more flexible coaching, “What can we learn from this? How do we make things better?” is going to be really important.

I, at one point, noticed that Google was struggling to hire SRE, so I made a pitch to one of the managers and said, “How would I build training program?” We’re going to reach out around Google. There must be loads of these smart people who aren’t SREs, but we’re going to turn them into SREs.”

Given the amount effort, we were expanding through the hiring process at the time, it made sense. WE built about a five-months full-time training program, went out looking for data center techs, salespeople, field techs, people doing internal IT who were really really sharp and have that kind of SRE spirit, this idea that, “I don't care what the problem is. I’ll work it out. I want to know full stack.”

My idea for a full stack is probably different to a lot of people. My idea of full stack starts at the hardware assembly language and goes all the way up into the understanding the customer space. A lot of people, when they talk about full stack, they mean Ruby and JavaScript, and that's not quite my idea.

Yeah, we give a five months full on training program and then about little over half of them moved straight into SRE. They would have had to do full SRE interview just like anyone else, and I kind of jokingly say that Google tend to hire about half of the people they should. Don't ever feel bad about trying to interview twice.

It in the fullness of time, about 85% probably ended up in SRE. The thing that they brought, it was — Yeah, that full stack understanding, that love of, “Here’s a software I’m looking after. It’s mix of C++, Java, Python, networking. We’re relying on 25 dependencies,” and those people have the technical depth to be able to understand every single one of them.

Yeah, when you move them on to a new team, they might go, “Oh, it’s a few years since I’ve looked after Java and now I’m looking after Java-based pipelines. , I'll brush up my Java. I’ll understand how the pipeline works,” and either wide enough technical understanding of Google in general for instance to make tweaks to the pipeline. You might have too many — You’re

kicking off a map reduce that's too big. Yes, 5000 looks bigger than 1000, but I know that that cluster is smaller so let's throttle down to 2000, and hey, we're 2000 shards. It's going to make more progress than 5000.

These kind of very very deep understanding, so an SRE who can go deep, but then has a wide understanding of the whole technical environment they're working in, that's really what they're looking for, having managers who can cultivate that are hard to find.

[0:37:55.1] JM: Okay, you mentioned something interesting there about — I think you said Google hires one half of the engineers that they should. Is that what you said?

[0:38:05.1] JL: Yeah. I think, historically, I would've been on many hiring committees and done a few hundred interview, and I suspect that, yeah, the hiring process is set up such that we're really good at definitely discovering we should not hire this person and we are —If they're weak, we'll definitely find that out, but because of that, we probably have a lot of false-negative where these are people who are really good but we don't hire them for various reasons.

Sometimes we don't want to waste a huge amount of time interviewing, so ideal thing is you give someone 50 interviews and they do well at 40 of them, yeah, let's hire them. Anyone can have three or four bad interviews in their life, and if you have them all on the same day, we won't hire you. That's kind of sad, but we're not trying to optimize for perfection. There's a certain amount of efficiency in there as well.

[SPONSOR MESSAGE]

[0:39:03.6] JM: Dice.com will help you accelerate your tech career. Whether you're actively looking for a job or need insights to grow in your current role, Dice has the resources that you need. Dice's mobile app is the fastest and easiest way to get ahead. Search thousands of jobs from top companies. Discover your market value based on your unique skill set. Uncover new opportunities with Dice's new career-pathing tool, which can give you insights about the best types of roles to transition to.

Dice will even suggest the new skills that you'll need to make the move. Manage your tech career and download the Dice Careers App on Android or iOS today. To check out the Dice website and support Software Engineering Daily, go to dice.com/sedaily. You can find information about the Dice Careers App on dice.com/sedaily and you'll support Software Engineering Daily.

Thanks to Dice for being a loyal sponsor of Software Engineering Daily. If you want to find out more about Dice Careers, go to dice.com/sedaily.

[INTERVIEW CONTINUED]

[0:40:17.0] JM: I think what was laudable about Google in — I just remember applying. I applied to Google so many times. I think I'd went through a bunch of different interview loops and I never made it, never got an offer from Google, and I would always walk out of the process feeling like, "You know, this is just so stupid," because I would know a person and I'd be like, "I know I would be a great fit at Google. I know I would do a great job."

Your hiring process — I would look at the interview questionnaire. You'd be something — Some interview question are like, "Oh, you've got a directed acyclic graph and at every node you've got a game of chess and you've got to find the best move in the game of chess at each node in the graph and you've got to solve all the games of chess at once and you've got to write an iterator to iterate across the graph. Actually, you're iterating across an iterator of iterators of the graph and you've got to write this in 30 minutes. Go."

I would just be like, "Why are you giving me this stupid problem?" I mean that's how they did it and they wrote a lot of information about why they did it that way. They were quite transparent about it. They were quite transparent about the fact that, "Hey, we bias towards no. We get a lot of false negatives and it sucks, but is the way that we do it," and it works, and it worked for them. I think, however long, 10 years later, or 12 years later since Google started talking about their hiring process a little bit, hiring is no less superstitious. It's no less of a black art. Nobody solved it. It's clearly just an incredibly hard problem to figure out.

[0:41:56.7] JL: It is, and I would say that in the early days, the hiring process was not as scientific. It was not as well calibrated. It was not as fair. The current process, some people might not like it, but it is fair. Everyone gets the same hard questions and they're all graded around the same.

In the early days, yeah, someone would write five minutes —Spend that interview and then write up like 30 words saying, "Yeah, this person was awesome. I think their code is great." These days, you have to write an essay and say, "No, this is why they were good. This is where they were weak. This is why I think we should hire them." its way more fair, more scientific. Google definitely higher less interesting people.

There were people that I can't believe they ever got hired, but they added so much flair to the team. They would be the — Not quite the mascot, but yeah, we would have hired an awful lot of people that had far more energy and maybe life experience than specifically hard engineers. That great manager that I mentioned earlier on, he was an anthropologist before he was an SRE. It's a lot harder these days to get past the Google interview bar as an anthropologist than it used to be.

[0:43:06.0] JM: You mentioned spending some time on the ads quality team, the ad serving team, and I've done much shows about advertising. I've probably been on the harsh side of things, but I know that building these systems is really hard and, actually, this is technology that needs to exist. It's really important.

Especially today, there are lots of deliberate quality issues. There are lots of deliberate fraud and scams, but back then, in the earliest days, I'm sure when you're just working on the ads quality team, the ad serving team, ad fraud was less of an issue than figuring out how to make the systems work. Can you tell me some about what was hard about that role? What was hard about doing ads quality and ad serving the earliest days?

[0:43:57.7] JL: I only transitioned into the ads well after that problem became well-recognized. Certainly, in the early days, it was mentioned, I think, in the company's IPO that solving the problem of click fraud was critical to Google's long-term survival. I think it absolutely was. If you go back 10 years ago, they assume that click fraud would end internet advertising. The

advertising companies worked out what was going wrong, what the click fraud was. They're never going to eliminate all of it because defrauding internet advertising is a multibillion dollar business, and any multi-dollar business, you can't stop it through regulation. It's like that war on drugs. There's just — If you take down some of the big players, more popup.

What you have to do is almost as cat and mouse game where the resources you throw at reducing the fraud have to be kind of in line with what you're losing because of the fraud. There's no point trying to stomp it completely because why spend $10 billion a year getting their rate of fraud from a billion to half a billion dollars for instance? No one really knows what the true rate of advertising is — Sorry. True rate of advertising fraud is, so it's very hard to any company to decide how much you're going to spend stomping that out.

One of my real concerns is that the industry now has very high bars for what it expects. Everything from people trying cloaking or fake accounts are buying AdWords accounts that would have been in use for a year by some business, and business decides stop trading, they could sell that AdWords accountant and suddenly this legitimate business can start putting in malware.

These are still very very hard problems, but only the larger companies seem to be able to throw the resources at solving them. I think a lot of that is behind — The last years, Google and Facebook taking the lion's share of all growth in that industry, and that's really concerning that if internet advertising becomes a kind of multibillion-dollar stakes poker game where you need to hire 5000 quality people before you can play, that's a concerning development.

[0:46:16.6] JM: I'm concerned that Google and Facebook and even solve it with hiring 5000 people. The main thing is like I don't — I'm not sure that brands — how aware are brands of this? Do they care about it? You're talking about; okay, Google even in the IPO, was saying click fraud is really important, but it's not even click fraud. It's like impression fraud. It's the fact that you can —

[0:46:41.0] JL: Yeah, you did an awful lot on the — I would've called it programmatic. What is it? Thid party exchanges and things like that. That gets very very hard to please, and especially — What was the other one? Where someone will set up a bot to pretend to buy something from

an online retailer then accept the cookie and then start sending traffic to their friend's publisher pages in the hopes of that retailer tries to come and pay a lot ad impressions. These are really really difficult. I think advertisers are not as stupid as we think. They do have a look at how much they're spending on Internet advertising. As that efficiency goes up or down, they change their spending.

I used to travel around a lot talking to some of our customers as well, kind of as a dog and pony show is the best way to put. Yeah, I was also really interested to see what our customers thought and some of the bigger players. They were aware of ad fraud but they were saying Internet advertising is still far better value than print, where they were spending 40%, say, of their advertising budget was on print media, but 84% of their customers time was spent on print.

They weren't considering optimizing that to be of great importance, and I think that could be some of what you see, that people are okay with huge amount of fraud if it's only proportionally a small amount of their spent. I assume as this changes and as the ad agencies understand more about what's happening and they get better at measuring it, we will probably see networks and agencies that don't take ad fraud seriously, just getting shut out of the market completely. That's the only way to solve this problem long term.

[0:48:35.7] JM: Right. The thing is advertising has always been about superstition and it's always been about these weird axioms like, "I'm wasting half of my advertising budget. I just don't know which half."
People still say that, like it is an axiom and that's the way that things have to be.

You hear that saying, you're just like, "That is insane. Who would be complaisant with, "I'm wasting half of my budget on X," especially in the days of big data where we should be able to measure so much of what we do. Advertising is the cornerstone of the Internet. We still can't really measure it.

I ask every person I have on the show who I'm talking to about advertising. I say, "Okay, what percentage of ads do you think are viewed by humans?" They have no idea. Like nobody has any idea what the hard numbers are.

[0:49:30.7] JL: We're not as far from the days of Madmen as you might imagine. Some of the more bizarre conversations I'd ever heard with a media buying executive. They were from a very large, say, food company and it was pointing at a lot of their competitors were spending far more on Internet and especially YouTube than they were at the time. I was trying to understand what was their reluctance. They came get back with something along the lines of, "You see, we understand print. We understand TV. If we want to run a three-week campaign, we can make sure the TV station won't run any documentaries on diabetes or any kind of health related content for those three weeks to avoid polluting our message. Anytime we ask Google to tone down, say, diabetes related search results, they've refused."

I was just speechless. I just could not believe that's how someone — I can't imagine — Yeah, I just couldn't imagine having a conversation like that with a Google account executive, "Can you turn off diabetes from the search results for a few weeks? We want to do a new campaign." This is one pattern of thinking in that industry.

Another one, I was doing kind of a presentation just on how data center's work and how their ads were being served. At one point, I mentioned that Google had turned paper mill into a data center, because people aren't using as much paper anymore. The reaction I got was almost violent where they said, "Prints not dead!" I was like, "What?" "Print publishing, it's not dead." "Okay, but it is."

In that case, they were a team who their account executives who were purchasing advertising, they'd come from the print business, maybe various magazines, and they wanted to buy magazine advertising rather than Internet advertising. If they're stuck there, then anything you tell them about ad fraud on the Internet, they're just going to say, "See? I told you, we shouldn't do any Internet advertising." Not we should be more smart about what we're buying, just told you we shouldn't do this. It's a very strange industry.

[0:51:35.1] JM: When I look at the biggest technology companies today, honestly, the two companies that are most impressive to me are Amazon and Google these days in terms of the giants. Do you agree with that assessment? Do you think Amazon and Google are kind of the two titans in the room? What do you see is the strengths and weaknesses of Amazon and Google?

[0:52:03.3] JL: I'm going to be bias because I have a huge amount of — A lot of fun memories of Google and really completely lived and breathed the culture for a very long time. I considered working for Amazon a few times. I think I interview twice and was kind of put off by the — They seem to run people very hard.

[0:52:21.3] JM: Lord of the flies.

[0:52:22.6] JL: Yeah, maybe. I've heard some terrible stories. I don't know which ones are exaggerations and which ones are true. Google certainly isn't easy, but the long-term institutional knowledge that long timers give to a company is so important. When I see I understand, two or three people where — Not everyone, right? But probably half of the people stay less than four years. That does give me pause on how fast they can adapt to changes later on. They have some great first mover advantages, amazing culture around let's not worry about making money on this yet. Let's just get the product out there, get users, get people excited about us, and that's more important than making money.

I think that's worked for them incredibly well. It's hard to see — My new employer, we use AWS quite heavily. I look around going, "Even if we wanted to move to Azure or Google cloud, it's not easy." There's a substantial amount of investment that millions of companies probably around the world have already made into Amazon. Yeah, that's a cash care for them long-term.

Interestingly, Amazon is not someone that I hear a huge amount of people say, "That's where I want to work," but they do, and a lot of people do, and a lot of people love it and learn a hell of a lot. it's working for them.

[0:53:51.8] JM: I think a lot of it is they've got a really good stranglehold on talent in the downtown Seattle area. There's a lot of people that want to live in downtown Seattle that are talented technologists, and Amazon is the biggest center of gravity there. You've got Microsoft. You've got Google that sort of have a remote presences there, but Amazon has a stronghold, and it's got network of buildings and —

[0:54:17.0] JL: It's still only one city. There are tens of millions of good software engineers out there. I think while having a big presence in a certain number of big hubs is interesting. I don't think it's the only thing. This this idea that you'll have a corner of a city that attracts load of jewelers, that's because people go there to buy jewelry and that's where new jewelers will open. Just by being a jeweler there, it doesn't guarantee that you're going to be successful.

I think Amazon probably benefited a lot from having Microsoft there in the early years because they could hire a lot of people in for Microsoft. Now, of courses, if you want to steal good people from Amazon, go Seattle, come to Dublin, and it's a good place to recruit. Those nodes of industry and technology fascinate me because we've noticed in Dublin in the last 10 years, it's all operations. There's not a huge number of large software engineering teams, but all of the big multinationals have huge data center and high-end operations teams. It's almost amusing. No friends who say, "Yeah, real software engineer, but I'm going to move out of Dublin because I don't want a job doing operations or SRE work. I want to be a mobile developer." They head elsewhere. London, for instance, seems to be a battles net hub of mobile software development. I find that fascinating how it just happens organically. Sometimes maybe governments try and kick it off, but those are giant organic nexus of developments in certain areas is fascinating.

[0:55:55.1] JM: You were around at Google in the days when Facebook was on the rise, and I've read a lot of stories about how Facebook was perceived inside Google and how Google was perceived inside of Facebook. I think this really came to a head around the Google plus days where you had the guy who was running Google+ was — Who wanted to run Google+ sort of whispered into Larry Page's ear that Facebook is a huge threat and we need to copy them and make — It ended up being a huge distraction for Google, and Google tried to integrate Google —

[0:56:32.7] JL: Yeah, okay. That's definitely a way to read it. It was certainly one of the times that I was proudest of Google was when a note went around one day from — Or I think it was talked externally about the earth quake. Yeah, Google was looking at Facebook, and their main perception was this looks like Facebook are building a walled garden. There's going to be lots of contents that we won't be able to index information and knowledge that will be locked away from humanity forever. We should do something about this. [inaudible 0:57:03.1] said, "Yeah." Put his hand up and said, "Give me 500 engineers and we'll start working on this on Monday morning."

There's a things in that. just this idea that I think Google+ was not a me too Facebook thing. It was how do we ensure that people have social discussions online, but not default to that being locked into a walled garden to protect some company's business and bottom-line.

The second thing, of course, is that Google, even then, 500 engineers been ripped away from other jobs and told, "Right, you got a high priority mission. Let's go." That was amazing thing to see happen. Traumatic, obviously, for anyone who would have been in a team with 20 people doing something like Google calendar or docs and be told, "Yeah, you're going to contribute a bunch of your team to this new team. High priority. Let's go. Let's go," and they did. In a matter of months, they did what Facebook took four or five years to do.

Ultimately, it didn't work. I think they're absolutely right to give it a go. Plenty of companies try and compete in an industry that there's a big first mover advantage and most of the time they succeed — Sorry. Most the time they fail, I think it's noble for them to try anyway. As it turns out, I completely agree with that. I used to have a personal website that did looked after Irish living history as well as discussions. We had a forum, things like that. Over the last few years, more and more people have taken to Facebook.

They set up closed communities there, because less hassle running website, et cetera. That made me sad because all of that content, those discussions, people taking pictures, giving advice, all of that's locked away from the search engines and it never gets seen again. Yeah, I wish Google+ had been a better success. I kind of wished — Actually, a better outcome would have been for Facebook to make Facebook more open, but that didn't happen.

[0:59:03.6] JM: There is this consumer desire to have more of a gradient of publicity versus privacy. You can't index everything on Snapchat. In retrospect, wouldn't the correct perspective have been to say, "Okay, people want different gradients of different levels of privacy. That's great. Facebook can have their thing."

[0:59:28.5] JL: Yeah. Imagine if every piece of content you had online — We had an X, kind of a standard for privacy. Something along the lines of, "I just written a blog post, share it with my friends or people who know my friends. It doesn't matter if it's Facebook or Google indexing or

your email, they all respect the ACLs of that items. I think it was just too complicated to do it back then. I wonder we'll be ever retrofitted on the Internet. Probably not.

[0:59:59.9] JM: Okay. I know we're running out of time here. The main point I wanted to make about the Facebook versus Google thing was looking at something as a zero-sum competition-base thing. That seems like what drove Google to make Google+, and it seems like it was ultimately counterproductive. Whereas if Google would've just said, "Okay, that's fine. Let's focus on our core products."

I mean is that an unfair lesson to draw from that, like don't focus on competitors. Just be focused on your product?

[1:00:33.3] JL: I think your absolutely right to say don't focus on your competitor, focus on your product. If you focus on the competitor, it's very hard to do better than them in anything. All you can do is catch up. Where I disagree with is Google+ didn't always know what it wanted to be. I'm pretty sure most the time it wasn't trying to just copy Facebook. It was, "Okay, if we say let's not copy Facebook. Let's try and do everything." Things like, "If you plus one a page, that could be like a shared bookmark. It could be great."

A lot of these ideas, they sound great in paper until you know plus one a website and then suddenly realizable, "Whoa! Do you mean all my friends know my browsing history now?" Then you try to explain, "No, it's not your browsing history. It's just that you plus one it and then suddenly various things break down."

I thought another huge distraction on Google+ was the real names debate, and this is where as an editorial decision, I suppose, they said, "No, let's discourage pseudonyms. Let's band fake names," because there was his theory that people using their real name would be more civil online. This is usually accurate, of course.

Soon, it turns out there's a lot of people who have very good reasons for not using their real name online. Some people have built brands around their name. Do you want to turn away Will-I-am because he doesn't have a real name. Do you want to turn away people who are blogging about domestic abuse survivors because they don't want their partner, X partner to find out

where they live and where they're writing now?

There's a lot of really really good reasons, and that was a huge event of energy wasted I suppose in those kind of debates. This was not a bad thing. It was amazing to see those debates happen. It was amazing to see the genuine passionate people out for it. It didn't work out, but I'd certainly don't think it was just because it was chasing Facebook and that wasn't going to work.

[1:02:28.6] JM: Okay, now you're at Intercom, and Intercom is a fast growing company. How does SRE at Intercom differ from Google? Maybe you want to quickly explain what the Intercom product is for people who don't know.

[1:02:43.1] JL: That's a good idea. I flippantly called in the early days, Intercom was kind of like Clippy for the web, but anyone who uses a lot of software as a service products will see a tiny little Intercom logo down in the corner.

[1:02:53.5] JM: You probably — You should not join the marketing team.

[1:02:57.0] JL: No. No. Definitely not. Do not put engineers in marketing. Yeah, you might see a little intercom logo down in the bottom right corner. The initial product was a way of contacting websites saying, "I have a problem," et cetera, and then the website reaching outside saying, "Hi, would you like help?"

It turns out, having the little icon on all the webpages allows us to collect an awful lot of intelligence on people's users. We can then combine that information with the information that the company has already gathered, like purchase history or maybe level of competence with the product or city level location. Then later on, design marketing campaigns like, "Hey, we went to — If someone arrives at our website and they bought something more than a month ago over a hundred dollars but haven't bought anything in two weeks. Use the information about their city codes and say offer them a 25% discount for the next 12 hours on some part." Then they can measure the impact of the campaign on their user-base.

We're doing a lot of really cool stuff allowing people to offer a very high-end customer experience as well as some pretty cool marketing tooling. It's early days yet, but certainly we do not have an SRE team. We've accidentally hired two SREs from Google. There's a bunch of X-Amazon and Facebook production engineering people. We know how SRE is supposed to work, but it turns out SRE and Google and SRE and Intercom would be utterly different things.

[1:04:28.1] JM: That's right. You're a principal engineer. Sorry. I wrote down your role wrong here.

[1:04:35.3] JL: I'm going to call it like a product engineering. In Intercom, they just want engineers who are focused on the product and making the product go, and while I'm playing around in infrastructure, everything is — In the Google mentality, I would be thinking, "Oh, the elastic search has to have an SLA. We have to meet it. We have to be able to automate all upgrades and we have to make sure that Mongo databases are flawless and beautiful hygiene and have all intent-based infrastructure as a code. In the Intercom, they'll look at you funny and go, "How does infrastructure as code help our customers?"

You go, "That's true. Yes. Okay, let's just let a lot of that slide and let's concentrate on the small things that make the database more performant." Or one of the big problems we have, of courses, is we'll have more and more customers who'd say, "Oh, great! Can I send an email to my — Or I could match this message? We have 10 million users or 100 million users," and suddenly you go, "Okay, I now have different priorities. My main priority is just building a data infrastructure that can take these kind of crazy arbitrary queries designed by product people who are looking to keep customers happy, rather than having a beautiful pristine production environment.

[1:05:43.8] JM: This is why I will always be a worshiper at the church of Google, because this is effectively Google infrastructure for everyone being manifested. Because at Intercom, you can just focus on product engineering. The reason you can do that is because — I mean, largely. Whether you want to blame Google or just the way that computer sciences has evolved, you have Google infrastructure for everyone. I'm sure there's plenty of technologies that you're leveraging at Intercom whether they're visible or they're beneath the surface that were Google papers back in the day and they made their way into the public infrastructure. Now, not

everybody can leverage them without even thinking about it and you can build a service that allows millions of businesses to spin up chat debates — Basically, chats on-the-fly from their webpage. That would have been unfathomable 10 years ago. How on earth would we —

[1:06:45.7] JL: Absolutely. I'm still suffering from culture shock. In Google, we build everything ourselves. Like everything, host, the phrase someone used. In Google we don't just —We didn't just reinvent the wheel. We vulcanize our own rubber. While in Intercom, there's a very much a, "No. No. Buy it. Don't build it. If you can buy it, buy it."

I think my first day, I had to sign up for 30 different software as a service, whatever login pages, because it was like "Oh! We have this CI thing. We have this on GitHub, and this on AWS and we'using various bits and pieces from all over." it was dizzying, and I have to say it was really unnerving because — It kind of feels disjointed. I know there are a lot of great things, "Oh, GitHub have Slack integrations, for instance." In Google, we had one build system and everything worked with that. We had one communication system like IRC. Everything worked at IRC.

In the SAS world, a lot of things kind of feel junky. They almost fit together. On the plus side, you can have a small company with a small number of engineers and have no problem deploying phenomenally enormous complicated databases. If you get a customer spike, you just say, "Oh, that's fine. We'll just add another 50 workers to that fleet, and it's fine." This just blows me away that it's possible, because that was the single biggest change in the last 10 years, was just how efficient startups have become and how that multiplier from, "I've got an idea and a small bit of skill and really good business sense," and how you can use the world's cloud computing infrastructure and software as a service culture to make that happen in months.

[1:08:30.1] JM: All right, John. It's been great talking to you. I have a lot of great stuff. Maybe we can do another show at some point in the future. I'd love to talk some more about Intercom. Anyway —

[1:08:41.4] JL: I think it would ultimately be really good idea to grab one of the more senior engineers who can do some good stuff in that respect.

[1:08:48.7] JM: Definitely. Okay, cool. Thanks for talking. Thanks for being a person in the Slack channel who is fun to chat with. I'm sure any listeners who want to hear more about your stories certainly can jump in the Slack channel and send you a message. Thanks, John.

[1:09:05.2] JL: Thank you very much, Jeff.

[END OF INTERVIEW]

[1:09:11.6] JM: You have a full time engineering job. You work on back-end systems of front-end web development, but the device that you interact with the most is your smartphone and you want to know how to program it. You could wade through online resources and create your own curriculum from the tutorials and the code snippets that you find online, but there is a more efficient option than teaching yourself.

If you want to learn mobile development from great instructors for free, check out CodePath. CodePath is an 8-week iOS and android development class for professional engineers who are looking to build a new skill. CodePath has free evening classes for dedicated experienced engineers and designers. I could personally vouch for the effectiveness of the CodePath program because I just hired someone full-time from CodePath to work on my company Adforprice. He was a talented engineer before he joined CodePath, but the free classes that CodePath offered him allowed him to develop a new skill, which was mobile development.

With that in mind, if you're looking for talented mobile developers for your company, CodePath is also something you should check out. Whether you're an engineer who's looking to retrain as a mobile developer or if you're looking to hire mobile engineers, go to codepath.com to learn more. You can also listen to my interview with Nathan Esquenazi of CodePath to learn more, and thanks to the team at CodePath for sponsoring Software Engineering Daily and for providing a platform that is useful to the software community.

[END]