EPISODE 638

[INTRODUCTION]

[0:00:00.3] JM: Blogging is more than 20 years old. Over that period of time, numerous publishing platforms have been created. Squarespace and Blogger and Medium and Twitter are popular closed source blogging platforms. WordPress has been the most popular open source blogging platform and much of the internet, including Software Engineering Daily runs on WordPress..

WordPress is powerful and news companies, e-commerce, websites and many other kinds of businesses use it as their central publishing tool. WordPress has been around for 15 years and there was room for a new blogging platform. There are also potentially some conflicts of interest between WordPress, the open source project and wordpress.com, which is a company started to host WordPress websites by the creator of WordPress.

John O'Nolan was working as a WordPress developer when he decided to start a new publishing platform called Ghost. Five years after starting Ghost, Ghost is a success with a thriving open source community, profitable SaaS business and companies like DigitalOcean and Mozilla using Ghost to host their blogs.

In today's episode, John and I discussed his background with WordPress and what he wanted to do differently with Ghost, as well as the software architecture of Ghost. We also touched on the ghost SaaS business and the management of the open source project.

Before we get started, we are also hiring a creative operations lead. If you're an excellent communicator, you can check out our job posting at softwareengineeringdaily.com/jobs. This is a great job for somebody who is an engineer, but they're looking for something that is more creative, more editorial, or somebody who just graduated a coding bootcamp, perhaps somebody with a background in the arts who's making their way into technology. I know there's a lot of people like that. If you want to be creative and you want to learn more about engineering, you can check out this role at softwareengineeringdaily.com/jobs.

[SPONSOR MESSAGE]

[0:02:14.8] JM: At Software Engineering Daily, we’re always analyzing data to determine what our listeners care about. We actually have a lot of data, even though we're just a podcast. It always reminds me that organizations with much more engineering going on have an order of magnitude more data, than a podcast like Software Engineering Daily. That's why the job of data scientist is such a good job to get.

Flatiron School is training the next generation of data scientists and helping them land jobs. Flatiron School is an outcomes focused coding bootcamp that offers transformative education in-person and online. Flatiron School’s data science program is a 15-week curriculum that mixes software engineering, statistical understanding and the ability to apply both skills in real-life scenarios.

All of the career-changing courses include money-back guarantees. If you don't get a job in six months, Flatiron school will refund your tuition and you can visit their website for details. As a Software Engineering Daily listener, you can start learning for free at flatironschool.com/sedaily. You can get $500 off your first month of Flatiron School's online data science bootcamp, and you can get started with transforming your career towards data science. Go to flatironschool.com/sedaily and get $500 off your first month of their online data science course.

Thanks to Flatiron School for being a new sponsor of Software Engineering Daily.

[INTERVIEW]

[0:04:06.4] JM: John O’Nolan is the Co-Founder and CEO of Ghost. John, welcome to Software Engineering Daily.

[0:04:11.0] JO: Thank you very much for having me.

[0:04:12.3] JM: Blogging and personal websites are a large part of the internet and many people use WordPress, including myself. WordPress is an open-source content management system. You can use it to make blogs and websites and much of the internet runs on

WordPress. Before you started Ghost, you were a WordPress developer. What are some of the problems with WordPress?

[0:04:36.8] JO: I think the pros of WordPress generally outweigh the cons in the macro spectrum, which is that you can use WordPress for just about anything and you can really do an incredibly diverse range of things with it as a piece of technology, despite a lot of the stick it gets. I think my criticisms of WordPress are in a really specific niche, which is for publishing, for doing specifically publishing, almost where WordPress really started out in that blogging era.

It has become quite cumbersome, quite difficult, quite slow and really moved away from that focus as its diversified and grown up, as you will. At the time, some five years ago when I was working on WordPress, I also was doing a lot of freelancing for clients and they all wanted blogs. They were all publishers. I felt the frustrations from a publishing mindset more than anything else. That's where I think WordPress these days is not really the best tool anymore.

[0:05:26.4] JM: When did you start having the idea to make a different platform?

[0:05:31.4] JO: I was contributing to WordPress for a good, I think three years, and probably two of those years I was thinking, there's space here for an alternative. I always rejected that idea in my own head as being too obvious and who wants yet another blogging platform? Was the general narrative and still is to this day the general narrative. I always thought that idea will never work.

[0:05:52.6] JM: Who was it that did want an alternative blogging platform?

[0:05:56.4] JO: No one outright. I think in 2013, there was this – we had this little mini-Renaissance of blogging, where a medium was just getting started, Svbtle was really — as an SV, Svbtle was really cool at the time. It does some [inaudible 0:06:08.5] thing. Goes to just had a blog post at the end of 2012 that went wild on Hacker News.

There's these little mini-resurgence of people being interested in blogging technology as a whole. I wouldn't say there's any specific group that was really demanding a new product, but there was definitely a renewed interest in the space again.

[0:06:28.1] JM: Well in software, there's often a pendulum swing between when a platform gets too complicated, then it creates an opportunity for something in the same space to emerge that is fulfilling some of the same use cases, but is more opinionated. It seems like blogging was such an expansive area that the fact that WordPress was dominating it, despite the fact that WordPress is not only a blog, it's a multi-purpose CMS and e-commerce thing and all these other things, it's clear that there was space for something more opinionated.

[0:07:05.1] JO: Yeah. I think that's fair.

[0:07:06.7] JM: What about the plug-in ecosystem? WordPress has this vast ecosystem of open source plugins. In some sense, that creates a network effect that's really hard to penetrate. What was your thinking on the plug-in ecosystem when you were going into Ghost to start it?

[0:07:22.3] JO: That's a really interesting one, because WordPress has this enormous plug-in ecosystem, which is more often than not, heralded as its biggest strength. It's a bit of a double-edged sword. There are a lot of good plug-ins for WordPress that can make it do all kinds of different things. There's premium ones which will add entire bits of functionality like an e-commerce store, like WooCommerce for example, and there's free ones which will just make WordPress work the way it really should have worked out of the box, like Yoast SEO plugin, for example.

There's a handful of them that really everyone uses. Every single WordPress site gets set up in the exact same way with a handful of the exact same plugins. Our initial take on that was why should you need plugins to do a lot of these things? Doesn't everyone just want SEO to work? Doesn't everyone just want their images to work correctly? Shouldn't these things just be baked into core? If you do that, if you have a sole use case focus of publishing and you build in all of the traditional plugins that people would have to install separately from the start, then well, you really negate a lot of the need for plug-in functionality.

Not to say that plugins have no purpose, but the purposes is vastly reduced versus WordPress. If you look at that plug-in directory, a massive amount of it is duplication. Thus, if you search for, I don't know, Twitter plugin for WordPress, you'll get a ridiculous number of results. Most of them are bad, there's one or two popular ones. You'll also get a lot that have really stagnated.

WordPress is starting to do a better job of hiding the old plugins from the directory, but there's tens of thousands of plugins which are not compatible with the latest version of WordPress, they have enormous problems when you activate them when – with conflicting with other plugins. It's not all plain sailing. Not to mention that WordPress's biggest criticism in the developer ecosystem by far and anyone who's ever heard of WordPress will have heard of the security concerns and the exploits. 100% of that is down to plugins, so it's not core.

The core team are pretty good at keeping WordPress tightly secured in every sense of the word, but they can't account for vulnerabilities in plugins, which expose an entire site. Yeah, I think there's positives to that giant ecosystem, but there are sufficient negatives that it's not as clear-cut as it might seem on the outset.

[0:09:42.3] JM: Right. Regarding that security issue with the plugins, I had a plug-in that was just serving some function on my WordPress. I don't even know what function it was serving, but then all of a sudden one day it started serving ads for acai berries and reverse mortgages and those salacious terrible ads that you sometimes see on these WordPress sites, because it was running some code that I didn't know, to do something like resize an image, or something stupid like that.

[0:10:13.3] JO: Great.

[0:10:14.6] JM: Yeah, thanks. What do you do in terms of a plug-in ecosystem? Do you have a plug-in ecosystem, or do you just try to bake in everything agnostic of plugins?

[0:10:25.1] JO: When we started out was we have in 2013 two developers and we're just trying to build the core platform and not think about plugins yet, because it's too much responsibility. Then we got stuck in that rut, and five years later we still haven't really figured it out. I think, we've got more clear-cut ideas now of what plugins are and what plug-ins mean. I think, that's also changed somewhat in the architecture of the web as things have grown and the way in which we build applications.

Currently, there are ways of extending and integrating Ghost using either Web Hooks, or Zapier. We try and look at the use cases for where people want plug-ins, what they actually want them to do, and usually it's very vague and a lack of clear use cases is the main thing that prevents us from doing something more urgently. Yeah, we're trying to figure it out.

I think what's coming up that's really interesting is this whole notion of serverless and lambda and the slight change in the architecture of how applications are built, where there's this interesting possibility of could you have plugins, or pieces of code exists externally to an application that would allow it to be extended without needing to be directly integrated into a codebase. I think that has some interesting possibilities.

On the whole, our biggest hesitation with just making plugins for the sake of making plugins has been as soon as a third party developer start depending on things, well then, you're in the backwards compatibility hell hole, right? You either go the WordPress root of you never break any third-party developers stuff ever and get stuck with a lot of baggage, or you go the Drupal route, where you break everyone's backwards compatibility and destroy your entire ecosystem every two to three years and start over. Neither of those seems super appealing. We've been hesitant, more hesitant about getting it wrong. Perhaps then we should be, but yeah, it's an interesting one for sure.

[0:12:17.5] JM: Let's go a little bit deeper on that serverless idea, because you're basically saying that if you wanted an ecosystem of plugins, it would be cool if they were maybe serverless functions that were deployed on AWS lambda, or Azure functions, or Google functions and these things are super cheap for execution. You could make some shim over that that either deployed these "plugins," basically these blobs of code to serverless functions, or you could – there's also these container instances which are longer longer-lived serverless instances, but then you actually have to keep the server running, so those can be a little more costly. What would that look like? Have you started to think more clearly about that, or are you very much in the ideation step?

[0:13:05.3] JO: We're in the stage of looking at that as a probable trajectory for the way in which the web is going in the next five years, and thinking about how we could fit into it as opposed to try and work against, or how we could take advantage of these new technologies that are coming, rather than be scared by being taken over by them, which is obviously going to happen to everyone sooner or later, so you either go with it or you go against it. There's only one way historically of those two that has worked.

It's taking the long view. I think, there's very little certainty on how that will work, what the economies of scale might be in terms of cost, but I think it's curious to imagine a plugin as a service-based architecture, if you will, where you perhaps have an open source, very small code base that lives in a centralized location, updated by a single developer, as opposed to needing to push updates in a more zip a package development workflow, but can do continuous integration on a hosted plugin living somewhere externally and have a decentralized network of sites depend on that functionality, and be able to maintain and deploy updates to it. I think that's really interesting.

Of course, it could still be open source. There's no reason that code has to be centralized in one location. It could be centralized in multiple locations, but that's one of the examples of where I think serverless or lambda is going to start creating interesting ways of building apps that we haven't considered possible. Well I mean, they haven't be possible until now, but we'll flip some thinking on its head potentially in the future.

[0:14:34.5] JM: What's an example of a plugin? When you're envisioning this serverless plugin ecosystem, what's an example of a plugin that comes to mind that could be quite useful if it was deployed that way?

[0:14:45.6] JO: You can ingest some data on your site, send it off to in an API request, send it off to a service to be processed or to do something with it or to be stored and then get a result back that you want to be rendered. I mean, that can go anywhere from as small as ad rendering, all the way through to maybe a full comment platform. It just depends what the architect the data and the users ends up being and how those pieces interact with one another.

I think, it's a little too difficult to form fully fledged examples, or fully thought-out ideas of how these things might play together, because there's too many unknowns right now, but I can I can definitely see the way in which people are using – if you think of it as extending the way in which people currently interact with headless CMS's and everything being an API and then go a step further beyond that to every piece of functionality being an API. I think you can follow that to a really interesting logical conclusion, where everything talks to everything else and your end result is a combination.

[0:15:44.6] JM: Yeah. Intuitively, it seems like AB testing and rollouts of new features and other kinds of testing and of course scalability, maybe even things like machine learning could be made significantly easier for consumer level, or the semi-technical use, or the type of user that's capable of operating a blog, but may not be a developer, giving them access to these tools in a way that's not preponderously expensive, or hard to operate that could really next level what a blogging platform even means.

[0:16:23.4] JO: Yeah, definitely. We're certainly thinking outside of that label. I mean, we have been for a few years now, the word blog has become somewhat uncool. I don't know if you've noticed, but people don’t really use it with the same feather –

[0:16:35.0] JM: It’s news to me.

[0:16:36.0] JO: - with which they did in 2008. Yeah. No, I think blogging is a concept, like a pre social-media concept almost is on its last legs, but publishing is not going anywhere anytime soon and there will be different models of publishing and different setups for that and different things to do with that long into the future. I think that's agnostic of traditional reverse chronological feed of posts. It can be something that looks very different that when I think of what's a modern newspaper, I think of something like Productan or Nomad list. They're still curated homepages of information, where you go to inform yourself about topic to see what's new to understand an industry, or a world that you're a part of a little better, this community participation, you can subscribe to the whole thing or different parts of it. I think those are really good examples of what modern journalism could look like that is far beyond the traditional written blog.

[0:17:32.5] JM: Let's tell a bit of the story behind Ghost. You were working on WordPress in some fashion, show what you were doing in the WordPress project, but then you eventually decided you wanted to start Ghost. You funded the project through Kickstarter, now you have a foundation, the Ghost foundation. Ghost is a non-profit, but you also have a SaaS business. You've got a unique financial structure of Ghost. Give a little bit of the history of Ghost and then catch us up to the present.

[0:18:05.0] JO: Yeah, sure. Years ago, I think I mentioned this briefly already. I was a freelance web developer and I started getting involved contributing to WordPress, because I felt it would be a competitive advantage to be on the inside of WordPress if I were servicing clients doing WordPress work, right? It seemed like an obvious thing to do to me, so I just started volunteering in the design working group of –
[0:18:26.6] JM: Wait. When you started working on WordPress, you already knew you were going to create a competitor?

[0:18:31.6] JO: No. I meant, a competitive advantage with my clients. Like if I could say I work on –

[0:18:35.2] JM: Okay. I'm sorry, I misunderstood.

[0:18:37.4] JO: I work on WordPress, then that would – that would seem to me, it would look pretty good to the clients I was selling work to.

[0:18:44.5] JM: Okay. Got it, got it.

[0:18:45.9] JO: Back in 2010 or something. They had just formed a design working group to make UI decisions within WordPress; still very new. There were only a handful of people involved. I shared up a lot. Then she got promoted to deputy head of that design working group leading a lot of the design ideation, interaction design and front-end development of those features from I think it was WordPress 2.9 through to about 3.2, 3.3, maybe 3., 2. something.

That was the initial route into open source and what got me hooked on those things. I also found a lot of things about WordPress, which I didn't like, where I felt could be better, or I thought about them in the manner of if I was to do my own thing, I would do this differently. One of the biggest ones was business model, as you mentioned. I saw WordPress have this this confusing relationship between wordpress.org and wordpress.com and Automattic and Audrey Capital and the WordPress Foundation, which are ostensibly five different organizations all controlled by the same person, but all of which have conflicting interests and a lot of overlap and confusion, to the point where even the tech media can't ever choose the right one when they're writing a story.

I saw WordPress becomes very popular open-source projects, and at the same time there were a bunch of these managed hosting companies which we're starting to do very well. There's WP Engine, you've got Flywheel, you've got Pagely, there's some other much bigger ones. That seemed a really obvious business model. You make the open source software and then you sell the manage hosting for it, which is a high price service. It's not shared hosting. We're not talking about GoDaddy here. We're talking about $30 to $40 a month starting point.
My idea for the business model was what if you just combined those things, and rather than having this conflict of interest between a venture backed organization on the one hand, which had priorities, which were overriding the open source, nonprofit organization on the other hand, why not just have a single nonprofit with a sustainable profitable business model doing the hosting and reinvest all of the money of that into developing the open source software? Completely clean. No mess. No like, "Oh, there are actually some VCs in the corner, but we hide them." Just a clean, honest business model that would be very transparent, a lot more like Mozilla than automatic." Wouldn't that be nice? That was part of that initial blog post I wrote of like here's an idea for our products and also an idea of a business model that could support it. Yeah, I guess people seem to like it.

[0:21:13.6] JM: With WordPress, what are the conflicts of interests that create that cognitive dissonance you're suggesting?

[0:21:22.4] JO: It's a lot of things. I think you see it a lot less as a bystander and you see it a lot more as soon as you start getting involved in the WordPress community, particular in the core community. You have Automattic, which has 350 million dollars of funding, and the CEO of which is Matt Mullenweg, who is the founder of WordPress who is also the head of the WordPress Foundation, which owns the trademark, but is not entirely linked to wordpress.org, which is the steward of the open-source projects, which is also owned and run by Matt Mullenweg.

There's a complete overlap in the leadership and decision-making of all of these things, but one of them has 350 million dollars' worth of expectations on it to return a unicorn-based valuation, or exit, or IPO at some point. The others, all of the others are marketed as this is to democratize publishing, this is to make the web better, this is for the people, and those things I'm sorry, no matter how much you say them, those things are categorically at odds. If you try to say you're trying to democratize publishing whilst overlooking the conflict of interest of 350 million dollars of old white guys' money on the line, then you're lying.

There's some real deception there. I think most people are not aware of it. To some extent, it's not the be-all and end-all of this discussion, because WordPress is open-source. It does have a GPL license and that does and gender certain freedoms, which are inalienable no matter what happens to Automattic, the company. In terms of how the leadership operates and how the product is built, it's very, very, very significant. Yeah, it really affects the trajectory of the product and I think you can see that in the development of WordPress over the last few years.

I don't want to be too negative on WordPress here. I think it's great product and it does important things. I'm not necessarily a fan of the product development strategy, nor do I think it's a very good one, but the product itself in the community around it is incredibly good.

[SPONSOR MESSAGE]

[0:23:30.0] JM: Azure Container Service simplifies the deployment, management and operations of Kubernetes. Eliminate the complicated planning and deployment of fully orchestrated containerized applications with Kubernetes.

You can quickly provision clusters to be up and running in no time, while simplifying your monitoring and cluster management through auto upgrades and a built-in operations console. Avoid being locked-in to any one vendor or resource. You can continue to work with the tools that you already know, so just helm and move applications to any Kubernetes deployment.

Integrate with your choice of container registry, including Azure container registry. Also, quickly and efficiently scale to maximize your resource utilization without having to take your applications offline. Isolate your application from infrastructure failures and transparently scale the underlying infrastructure to meet growing demands, all while increasing the security, reliability and availability of critical business workloads with Azure.

To learn more about Azure Container Service and other Azure services, as well as receive a free e-book by Brendan Burns, go to aka.ms/sedaily. Brendan Burns is the creator of Kubernetes and his e-book is about some of the distributed systems design lessons that he has learned building Kubernetes.

That e-book is available at aka.ms/sedaily.

[INTERVIEW CONTINUED]

[0:25:05.0] JM: I can absolutely see how the incentives are set up to be not completely aligned with democratizing publishing. Do you have any examples? I've used WordPress for the last 10 years, so I like feel I know how the WordPress ecosystem has evolved to some degree. Are there any particular examples that come to mind where you're like, that was an example where it would have gone differently if there would have been five independent people in charge of these five different organizations, instead of one person with cognitive dissonance?

[0:25:37.1] JO: Oh, there's so many. Yeah. A lot of it is very inside baseball, that's quite not interesting to most people who are listening to this. There's been a lot of odd decisions over the years around when to enforce the GPL to the fullest extent and when not to. It seems like when other commercial companies are able to make a lot of money out of WordPress, then that law, or some rules get enforced more heavily than others.

People get banned from speaking at WordPress conferences if they don't fulfill a certain version of the license, which doesn't apply to other areas. It's all very inconsistent, that's the main thing. In terms of product development, there's decisions, which I think if WordPress, the open source project was truly a democratized organization, or at least an independent one would have been made different in terms of what things are optimized for, but that's more speculation, I think.

I imagine, WordPress the open source project would have gone in some different directions if it were not for the implicit need to compete with Squarespace and Shopify and Tumblr simultaneously, which is the burden that wordpress.com bears the commercial organization.

[0:26:45.2] JM: Right. The licensing stuff, and I know it's inside baseball, but trust me, the listeners, they are fans.

[0:26:53.5] JO: Fans of baseball?

[0:26:54.9] JM: They are fans of the most inside of baseball games.

[0:26:59.7] JO: Well, as this is my favorite subject.

[0:27:01.6] JM: Let’s go there. License specifically. What are the things, if I'm a WordPress hosting company, like a WP Engine, or Hostgator, or Bluehost, one of these WordPress hosting services, what could I do to induce the ire of Matt Mullenweg and his five diverse organizations?

[0:27:24.1] JO: As a hosting company, not much other than perhaps use the word WordPress in a domain name. That right is reserved solely for Automattic –

[0:27:33.9] JM: Oh, that’s bizarre.

[0:27:35.0] JO: - and nobody else.

[0:27:36.1] JM: That's why all these things are WP blog.

[0:27:39.3] JO: Exactly, right. Yes. The trade doc is only –

[0:27:40.7] JM: WP happiness.

[0:27:42.4] JO: Exactly right. The trademark is not permitted for use by anyone in a top-level domain name. We do the same, to be honest. This is not entirely disingenuous. We just don't also then license it to a commercial organization that we own, or control. Drupal also does the same. As a hosting company, there's not that much that you can really do. Now, if you on the other hand are a theme organization, a theme company, or a blogging company, well that's where things get very different and that's why we have to start talking about the GPL and what the GPL means and where things get very interesting.

I tweeted a few weeks ago. The GPL is the communism of software licenses, and then that made a whole bunch of people really angry. Then as I did like, “Oh, I can't believe people who have – who make brush statements on Twitter with absolutely no context and think this is a suitable medium for discussion, me and then this tweet also me,” so someone hypocritical. The GPL gets interesting, because of the viral nature of the GPL starts coming into play when you when you talk about themes and plugins, so you have this core code base which is GPL license, the viral nature of the GPL says anything that is a derivative work of the original code also inherits the license.

Is a theme, or plugin, or an extension built on top of the core platform, is that derivative work? Now according to courts of law, specifically American ones, we don't know, because it's never been put to trial, it's never been put to a test, so there's no legal precedent to operate from. According to Matt and Matt's lawyers who were very highly paid, it means that well, essentially whatever he wants it to be, but if you try and build a theme for WordPress and you say, “I'm going to license this with a commercial license, then that's not allowed.”

You're banned now from all WordPress conferences. You're not allowed to speak there. I'm not sure if you're allowed to attend, maybe it's just not speak. You can't get listed in any WordPress plugin, repository or store. If you in any defy the GPL, then you're locked out. When it comes to licensing, I find that to be an obnoxious choice. I think that's the Jehovah's Witness approach to software licensing, where you say, you can come and be a part of this cool club. You can have this free software I've made. You can play with the cool kids and we're doing this great thing for the world. But if you join us, not only are you are then a part of our religion, but all of your kids that you have also automatically a part of our religion and there's absolutely nothing they can do about it. They have no say in that matter.

That seems a little odd to me. Isn't the whole point of open source to give people freedom, to give people choice, but at the same time you're then taking it away again? When we were thinking about licenses for Ghost, we absolutely did not want to go anywhere near that route. I think it's a much more powerful idea to give people the choice of what license to use and leave it up to them. We say Ghost is MIT licensed. If you want to build something on top of it and release it as open-source, that is you're inalienable right, if you want to build something on top of it and release it as close source or not release it at all, do something commercial with it. That is also your right, because open-source gives you that choice to choose.

The hope is that you would look at our example, us as role models and perhaps be inspired by it and also want to do something good for the world, also want to pay it forward in some way, but the notion that that is optional I think is more powerful than the notion that by being given something, you have to give away your future rights to your work as well.

[0:31:10.7] JM: GPL, that means that if they copy or if they make – if I'm a WordPress theme designer, I make a custom theme, I can't sell it as closed source software?

[0:31:24.8] JO: Correct.

[0:31:25.2] JM: I see. Now I see your communism analogy, because it forces everybody to share to the same degree that WordPress arbitrates.

[0:31:35.5] JO: Correct. Yes. It optimizes for the good of the people, rather than the good of the individual is the theory. I will say that everything I've just said is extremely nuanced and I have opinions and I don't expect everyone to agree with them. There are different interpretations of this. People who are GPL activists, or GPL fans will say that this is for the good of software and it's copyleft, which is the opposite of copyright. The whole point is to enforce the perpetuation of free software, so that it can't at some point be taken over by Microsoft and be taken away from the people.

There's different ways of interpreting this. Personally, I think if you want that approach, or if that's what you're worried about, then the best license is the MPL, and the Mozilla Public License protects against the original code from ever being changed. If WordPress was MPL, then WordPress core could never be relicensed as closed source. Anyone can build whatever they want on top of it, like that freedom is still granted, but the original code could never be changed. I think that's a cool model.

Problem with the MPL is if you say your project does MPL license, you're going to spend the rest of your life explaining what that means. That's also one of the reasons Ghost ended up as MIT, because after, I don't know how many years of GPL debates, I was pretty ready to focus on a project where I didn't need to discuss the license every 10 seconds. You just say, “It's MIT,” and everyone goes, “Cool,” and that's the end of that discussion.

[0:32:58.7] JM: Well, I think I can see what twists you up a little bit about the fact that it would be this communist style license, to use your words, when in contrast WordPress, the word cannot be used in a URL that is not wordpress.com. I sense that there's some hypocrisy from your point of view between the openness, or the forced openness of for example, themes, theme extensions, versus the overall mentality of the WordPress the company, which is by virtue of taking on a bunch of venture funding has to have some proprietary advantage over the vast populace of WordPress, other WordPress hosts. Without going further inside into the baseball, I think we kind of – that we get the point that there is some conflict in terms of the governance model that you recognize.

[0:34:00.8] JO: Yeah. I think that's a fair assessment. I'll say that I don't have any particular criticism for Automattic or wordpress.com needing to optimize for shareholders, or fiduciary duty that's what a company has to do that is the model. I think the summation of this entire discussion and the different avenues we've explored here is there's confusion, there's disparate motivations and interests and all of this culminates regardless of what specific issue we’re talking about and what the exact answer is to any of it. All of this confusion and disparate interest culminates into a community, a business model, an ecosystem, which is somewhat cloak-and-dagger; it presents itself as one thing, then you find out it's something else.

Then you really have to try and figure out what – okay, what's the real truth in here? What's the real motivation? What's the real story? That's a very, very frustrating experience. I can tell you the core contributors to WordPress, I became very disillusioned with it after a couple of years of contributing on an entirely volunteer basis, my time and my energy to open-source. Our model was as much inspired by lots of other economic and marketing-based reasons, as it was by wanting to have a business model, a structure, a license, something that was just transparent, incredibly easy to understand, and where you could quickly assess and understand the motivations and the interests behind decisions being made, which in our cases MIT license not-for-profit organization is just, we try and build the best product we can, that's it. There's not really any other weird stuff going on.

[0:35:33.8] JM: Speaking of weird stuff, WordPress is mostly PHP, I believe. Since you started Ghost, what three – was it three years ago?

[0:35:44.4] JO: Five now.

[0:35:45.3] JM: Five years ago, okay. You started Ghost five years ago, the front-end world had moved on so much that you could – I mean, again no offense to WordPress. They started in PHP, because that was among state-of-the-art back when WordPress was started, but you got a clean slate, so you got to choose your own adventure when it came to what front end you wanted to use. Can you describe the early architecture for Ghost and what you got right, what you got wrong, whether you've had to make any significant adjustments or refactoring since then?

[0:36:23.9] JO: Yeah, absolutely. I think the first version of Ghost, you could describe as a PHP application built in node. It was node. I'm being slightly self-deprecating here. I came from a PHP world. My co-founder and CTO Hannah was very much a PHP developer transitioning into JavaScript. Even just in terms of how the code base was structured, it looked structurally a lot more like a PHP application than a node application, just because that's the world we came from.

What's interesting is that, I don't know, I might have a slightly different answer here to what people might expect. I have very, very few criticisms of PHP on the whole. If anything, I think choosing to go down the Node.js route, or the full stack JS route has been equal parts, hindrance and help. There are some really, really interesting PHP publishing platforms out there at the moment, which are not WordPress and are not in the WordPress space; Craft CMS is one incredibly powerful, very interesting built on the E2 framework and then you have Statamic, which is very similar, a very similar product built on top of laravel. Laravel is one of the most awesome things I think that's happened in the last 10 years.

How do we gone down a PHP root? I think from a product point of view, we'd probably be a lot further along than we are now, because in the PHP ecosystem much like in the Ruby ecosystem, or even the Python ecosystem to some extent, this tooling is well understood. It has evolved over a long amount of time. You have your defaults that just work. You have your libraries, which have been time-tested, proven they're great.

Whereas in end of 2012, beginning of 2013 starting out into the node.js ecosystem, pretty Wild West. There's not a lot of stuff going on. There was barely a usable authentication library that we could start working with. Even to this day, it's not – there's a lot of things where if you were working in Ruby or PHP you'd go, "Yeah. Obviously, I'll just grab this library that everyone uses." Whereas in node you go, "I can't believe that doesn't exist."

I think when we wrote the first version of Ghost that wasn't a usable RSS library for parsing XML and delivering RSS feeds in node, and that was just mind boggling. Then that has repeated enough times to where it really slowed us down. What we got from that early choice to go full JavaScript was some marketing buzz, like if we had said, "Oh, we're building another publishing platform in PHP, unlikely that would have gained a lot of developer excitement or interest. On the flip side, it's been a lot more technically challenging, particularly as the Node.js ecosystem has also evolved more towards microservices as opposed to fully fledged web applications for lots of obvious reasons, which has also been a challenge.

It's been a real mix in terms of those early architectural choices and which ones paid off and which ones didn't. You're asking like what was a good decision? What was bad? I think one that I sometimes mentioned that we made early on was we built Ghost more all in one repository, more monolith style than decouples into modules, and that got a lot of stick from Node.js big-name developers early on, because they were like, "You're doing it wrong. Everything should be an NPM module, everything should be decoupled always, you should have 364,000 different repositories and then NPM and solve them."

We were like, “But we don't even know what the modules are yet. We don't know what makes sense as the module. This is a sandbox. We're trying to figure out what this app even is, what it does, where the moving parts are.” We, I say we, it's mainly Hannah, when we talk about the actual architecture here, said we're going to stick to our guns. We're going to stay more monolithic just to begin with, so we can figure out where the edges are, what things make sense to pull out into components further down the line.

That turned out to be a really good decision, because it allowed us to move faster, it allowed us to build more quickly, it allows us to deliver of product that we could then iterate on, rather than worrying about a long-term architecture for a product that was as yet unproven.

[0:40:27.9] JM: Okay. Two good lessons so far. One, I need to stop dissing PHP. I thought I'd learned this lesson, because I did an interview a while ago with the – I think it was the Slack CTO, or VP of engineering. Did you know Slack is in PHP?

[0:40:42.8] JO: Yeah, the backend, right?

[0:40:44.3] JM: Yeah, I guess it's just the backend. I think was Keith Adams, and he came from Facebook to Slack, so he came from the biggest PHP app ever to probably the second biggest PHP app ever.
[0:40:58.8] JO: Now Facebook is all PHP.

[0:41:00.6] JM: Right. No, that's what I'm saying. Yeah, he came from Facebook.

[0:41:02.5] JO: Oh, I see. Sorry.

[0:41:03.6] JM: Started working on Slack. Yeah, so anyway I got to stop dissing PHP. Okay, so when we start to talk about how to – Okay, now you've given the overview for the Ghost architecture, or given a little bit about it and you said – I thought the aspect of keeping it monolithic, the virtues of the monolith are well-taken and good to think about. There are other CMS's, like the headless CMS’s, like the Gatsby. We had Gatsby on the show a while ago and there's also Netlify, I think. What do you think of these models? Are these playing in – these headless CMS's, are these in a different space than Ghost? Do you think of them as for a different type of customer?

[0:41:49.1] JO: I think in many ways, they are the natural evolution of where things are headed next. I think they're very, very interesting, especially Gatsby, especially Netlify, especially a bunch of these newer platforms which are for now pretty small, are doing really, really interesting things. I honestly think that in five to 10 years' time, Ghost will look more like them, than them looking like us.

I think having apps be more decoupled, more API-driven, whether they are open source or closed source, whether they are centralized or decentralized is likely going to be the way we end up going. Yeah, I'm not sure of course, but I'm a big fan. Put it that way.

[0:42:30.2] JM: Yeah. I think in that podcast, we were talking before the show you did a podcast – you have a podcast, I think it's This Weekend Ghost, is that what it's called?

[0:42:37.8] JO: Yeah, we do it every six months.

[0:42:40.2] JM: Okay. All right. This is half, yearly half in ghost. The most recent one that I heard at least, you and your co-founder were talking about the API for Ghost and the fact that you can just treat your Ghost backend as a JSON API, I think there's – maybe you're still working on the making it permission, but it's clear that that's part of the vision. The idea of decoupling it from the frontend, so have people written custom frontends for the Ghost backend API?

[0:43:16.4] JO: Yes. A couple have. Not that many, but Ghost has always been a self-consuming JSON API, if you will. The Ghost core is a JSON API and Ghost admin is a client-side ember app that writes to it and I think that serves your site is just a little express server with handlebars that renders pages. It's always been designed with these three parts. Currently those parts are tightly coupled and what we haven't gotten to yet, but we're I very much see us going is in loosening those couples and eventually making them entirely optional, so that you could use Ghost as a headless CMS if you wanted to, either self-hosted, or centrally hosted, and you could build different experiences on top of it.

I think the most notable version of that that's currently happened is actually Apple. They use Ghost for all of their internal team websites, so they've built custom frontends for that, but they use Ghost as the primary platform behind it. I was talking to someone there and they said the one modification they made where they hacked core was to put in Apple's LDAP, SAML, whatever the hell it is, iCloud-based use the system rather than ours, which makes perfect sense. I'm like, I wish you'd pull requests to that upstream, because that would have been a handy thing to have, but no, cool to hear.

[SPONSOR MESSAGE]

[0:44:42.3] JM: Over the last two years, I spent much of my time building a video product. We had issues with latency, unreliable video playback, codecs. I was amazed that it was so difficult to work with videos. As it turns out, video is complex. Figuring out how to optimize the delivery of video is not easy, especially since there is both mobile and desktop and mobile users might not have as much bandwidth as desktop users. If you're an engineer working on a product that involves video, you just don't want to think about any of this. I can tell you that from firsthand experience, and that's why Mux exists.

Check out mux.com and find out how Mux makes it easy to upload and playback video. Mux makes video hosting and streaming simple. Today, you can get $50 in free credit by mentioning SE Daily in the sign-up process. Even if you aren't working on video right now, if you think you might work with video in the future, Mux is a really useful tool to be aware of. Check out mux.com. If you're an engineer who's looking for work, you can also apply for a job at mux.com.

On Software Engineering Daily, we've done two shows with Mux, and I know that Mux is solving some big difficult problems involving lots of data and large video files. To find out more, you can go to mux.com. You can get $50 in free credit by mentioning SE Daily, and you can apply for a job if you're interested in working on some of these large challenges. Thanks to Mux for being a sponsor of Software Engineering Daily.

[INTERVIEW CONTINUED]

[0:46:37.7] JM: I want to talk about the development and the open source nature, because much of that podcast that I heard between you and Hannah is the name of your co-founder, right?

[0:46:46.0] JO: That's right. Yeah.

[0:46:46.9] JM: Yeah. Much of that conversation, or a lot of it was the fact that open source is really hard and it's hard to manage this mob of people, sometimes the issue – your issue tracking can just get completely out of control, you've got to use Github, it's not like there's really alternatives to Github for social coding. Tell me about the biggest challenges of managing a large open source project.

[0:47:15.4] JO: It's difficult and I feel like it's getting harder. There's different models to this as well, so I feel if you are the maintainer of a library, a development library, which is the sole users of which are other developers, you have a somewhat different experience to if you build a consumer, or at least end-user product. The one has – the former has less opinions and more is more open to it's not – it's all code at the end of the day and most of the time, there are bugs that need fixing, but there's less opinions or stigma around the way in which that works, to some extent. I'm very much generalizing here.

Whereas, compared and contrasted to a product like Ghost, like any web app, or perhaps bootstrap is another example, when there's a visual component, there's a UI, there's a front-end, there's something to interact with, something to have an opinion about, then it becomes this people want features. They don't just want code, they don't just want the output of an API to change slightly, but they want and they have entire use cases about how things should work differently. You also attract a much broader level of interest and/or criticism, because it's not always just developers.

I think whereas, in the past where people use things like crack, I mean WordPress still does, you would have this barrier to entry, which would mean that people would have to work quite hard to get involved in open-source, and as a result, they'd be quite willing to be involved in the process of making open-source. Whereas now, with Github being a pseudo-social network, I think it's – and I mentioned this in the five-year blog post that we did recently. I think it's become too easy to demand support, or to make feature requests, but to bear no responsibility for contributing to open source whatsoever.

I always try to reinforce to some degree, and when people say this should work like this, or this should work like that, or why is there no PostgresSQL, which is the one we get most often, that if you use an open-source product, you are by default one of its maintainers. If you use an open-source projects, you are by default one of its maintainer. It is not up to some anonymous group of people working for free to sit around a table and fix the thing that you're outraged about. It is up to you to participate and it is up to you to get involved.

Github I think has made that worse, not better. It's too easy to open an issue. It is too hard to close one. It's too easy to get a whole load of reactions and thumbs up and emoji and comments on an issue and it is too hard to open a pull request still. In many ways, Github has become this transactional support tool for maintainers, which has become overwhelming in a lot of ways. I talk to a lot of the really big open source maintainers, because we compare notes on various points, how depressed we're feeling about people's negative attitudes towards open source as a whole and if there's anything we can do about it.

I've got to tell you, there's a lot of people who are in a really bad spot. There's a lot of maintainers who are running some of the biggest projects out there who don't have a sustainable business model, like we are very fortunate to have who get even more demands and issue requests on that time, and even less help. I don't feel Github have been working to solve that. I think there's a big wasted opportunity there in their open source creator community, which has not been capitalized upon in the way that YouTube do a really great job of collaborating with their creator community, Github don't.
It's tough, it's tough. I think, for the most part people still don't get it. When I say people, I include the vast majority of the developer ecosystem in that. I'm still not sure how we fix it and unfortunately, every keynote I've ever seen, every podcast I've ever listened to, every blog post I've ever read, particularly Noland Lawson's, the ultimate example, what it feels like to be an open source maintainer.

[0:51:06.1] JM: I have read that one.

[0:51:07.5] JO: Comes to the same ultimate conclusion, which is this is really difficult and I don't know how to fix it.

[0:51:13.7] JM: That post is so depressing.

[0:51:15.5] JO: It's one of the most depressing, because of how relatable it is. Yeah.

[0:51:19.1] JM: That's the one where the guy is like, he's made some popular open source repository and he just gets burned out, he never checks it and then he talks about going to check it after it's been two weeks. He goes into his notifications and just clicks on one, it's somebody flaming him and he's like, "Oh, my God. No. Just hurts. It hurts so bad."

[0:51:38.1] JO: Yeah, and fat from what's his name, real name? Jacob Thornton. One of the co-creators of bootstrap, works at Twitter and Medium, a bunch of other things. He's got a really great keynote from was it Law Web, or The Next Web, it was somewhere in France, so a few years back of the same thing, like Friday nights at home trying to close Github issues of people with false bug reports. His friends texting him and asking him to go out and him feeling really upset about that.

[0:52:06.3] JM: I bet Microsoft can improve that.

[0:52:08.5] JO: Well, this is the sad thing. After I wrote my five-year blog post reflecting on some of this and sharing some of the thoughts I had about Github, someone pretty senior at Github reached out and said, "We care about this and we want to work on it." Then about two weeks later, the Microsoft acquisition happened and I never heard from them again. That's not at all surprising to me. Unfortunately, I think Microsoft in the long run and the long term trajectory may well improve this, but in the short term trajectory, it's unrealistic to think that the next two years will be anything but a long handover a transition period, probably a lot of churn and headcount and a lot of integration work into existing teams and products.

I don't see Github doing anything for the next two years. In fact, I think Gitlab are in the most opportunistic position they've ever been in right now, because Github are going to struggle during this transition to make meaningful progress. We'll see. Short-term, I don't have high hopes.

[0:53:05.6] JM: Okay. A lot there we could continue to go into. I want to ask you a bit about the SaaS business, just because that's another area. I mean, there's so much meat to this project, so much we could discuss and I'm hoping we can get your co-founder on as well, because I want to talk to her, because this project is super interesting, but the SaaS business – so you have built the hosting platform, and I know that's not easy, because you've got scalability, you've got to pick out a cloud provider, you've got to scale a support team, you've got to figure out pricing. Building a hosting service is super hard. What are your hardest memories in building the SaaS platform and your biggest learnings?

[0:53:47.6] JO: Oh, God. Many. It's a little tangential, but it's very related. One of the things we've done well that has also really bitten us in the last five years is that Ghost has established itself pretty well as a mainstream open-source brand, or brand within the tech industry. A lot of people know what it is, a lot of people are pretty familiar with it. It's pretty high-quality in most of the areas that people interact with. It looks for all intents and purposes like a bigger company than it is. On the one hand, that's great. We get big customers, we get a lot of organic growth, we've never done any marketing whatsoever and revenue has always grown consistently since day one.

On the flip side, the expectations are exorbitantly high in many regards, and we run the entire hosting platform which serves over 100 million requests a month with currently one person, one part-time assistance administrator. We did have two, but team is currently down to one. The entire product team of the Ghost open-source product is only two fulltime people. We have this tiny team that is taking on disproportionately large engineering challenges, going up against competitors who I mean, the biggest two of whom have three – both have 350 million dollars of VC behind them and San Francisco's hipsters support network to help them do their thing. We're actually really, really, really small. That's not always apparent. That's not always clear. The building, I think platform has been a monumental task. If you had told me how monumental five years ago, I probably have considered the entire business model, because it turns out, which we discover quite quickly hosting Nodejs is also about 752,000 times more complicated than hosting PHP. Who knew, right? The most painful parts I think have been around learning that and understanding that and really figuring out how to build out a hosting.

[0:55:44.4] JM: Why is it so hard?

[0:55:46.2] JO: When you have PHP and Apache, you pretty much just run Apache and then you throw PHP on it and everything you do is just a PHP function, like it's the state of your dev environment is just there perpetually, or in perpetuity and works. Whereas in node, everything has to be initialized, everything has to be installed, you have all your build processes, then things are running, waiting on each other. It's just a very, very different world.

You hit scaling issues a lot more quickly and the complexity of maintaining that environment is significantly higher. For a more technically complete answer, if you have Hannah on the show, I'm sure you can draw those comparisons. Yeah, I mean, you can throw a whole bunch of WordPress sites on one Apache server and they're all essentially just work. There's nothing to do. You unzip the package, you put it in a folder, as long as that folder is mapped to some domain, it will run. Any Node.js app, you unzip it, you decompress it, you install the dependencies, you run the install process, you run the build pipeline, you hope that the output compiles, you run a CLI tool to configure the various different environment variables and get everything going.

Then you hope then pray that it all works. Of course, usually it doesn't. Whereas, in most cases for a PHP app pretty much just have one environment and that's how things are going to run. The ways in which people want to install node apps are very diverse. We say we will run the latest, we will support the latest LTS of node and we install guide covers, Ubuntu, I think 16. whatever at the moment. Then by and large, people will come along and say, “Excuse me, I'm trying to install Ghost on a Raspberry Pi that's being powered by three potatoes in my garage, using Center OS from 1992 before Center OS was invented, but also using some pearl proxy of JavaScript, rather than an actual JavaScript.

It's not working. Why is that? You're just like, “Why are you doing that? Why are you doing that?” It's wild, but people want to use their favorite thing. They always want to use their favorite thing. Those challenges are pretty big, but that's a little off track. In terms of hosting architecture, it's a lot more expensive, a lot more complex and requires a lot more work than I would have guessed. We started out on hardware in a datacenter in the UK, then we figured out that every time we'd need to scale, we'd have to order a server two months in advance of needing it from Dell and that didn't seem it made sense in 2015. We moved to the cloud. We have a partnership with DigitalOcean, which is great. They're amazing company and they've been very, very good to us over the years, and things have been fantastic working with them. That's gotten a lot better.

Yeah, but in terms of business model, it's panned out. It's allowed us to continue working on open source and that's the real purpose of it is to have a hosting company, which is able to be profitable to an extent that it can employ other people to not work on anything to do with the hosting company, as in to work on the product. That's the real magic of the business model, that's what the position we're very fortunate to be in, where we're able to do that, because obviously, a lot of our open source projects which are libraries, or frameworks, there's not any hosting you can sell for that, but we can, so that's been really, really good.

[0:59:01.9] JM: The great thing about the hosting business is people are sticky. If you get a customer, they probably stick around I still use Hostgator. No offense to Hostgator, but I still use it. Enough said. Also, I think the thing – the win that's at your back and is the same thing that's at the backs of Medium and WordPress and Squarespace and all these other hosting websites is there are so many more people who are coming online. Look, a lot of the – I don't know the statistics, but I think only a little more than half the world is online and some small portion of the people who are online have enough wherewithal to, or the desire, or the resources to set up their own website.

The number of people who have the wherewithal and who have the resources to do that is just going to grow. The pie is just getting so big, and I think you're pretty well positioned to consume your fair share of it, so it's a good place to be. I know we're out of time here, but I wanted to ask you just one more question. Long-form content on the internet, some people use medium, some people read whatever random WordPress blog, some people read Kindle books, I read long-form content on pocket, I really like pocket. How is long-form internet content consumption changing?

[1:00:22.5] JO: Wildly unfortunately. I largely don't think it's for the better. I think all the posts about the death of the open web, which apparently no one ironically notices are appearing on Medium. Don't go far enough. I think they're all completely accurate and it does worry me and does scare me somewhat. I think the way in which we used to 10 years ago visit each other's personal websites, or blogs, or company websites, or follow a broad range of our own idea of what a social graph should be, or a web graph should be across RSS feeds or anything else has largely died and it hasn't died, because the tools for doing those things have stopped being good, or have stopped working. It's died because of the proliferation of social networks and the addictiveness of all of all these things in terms of where we spend our time.

It's not that the rest of the information we used to go and look at or read doesn't exist anymore. It's still there, might be diminishing, but it's so much more of an addictive experience to log on to Twitter, or to see those likes racking up, or to go onto Instagram and do a duckface and get your social hit of dopamine from the likes on that.

The way in which people interact with each other on the web has moved very much into centralized silos, where long-form either doesn't happen as much, or if it does happen, it's not truly open, it's not truly happening in the same way. I have a lot of worries around that. I mean, I know this is – it's not like I'm making some broad revelation. This is pretty popular topic at the moment with even Apple and Google and everyone admitting that there are devices become too addictive and whatever the features they come up with, gold conscious time, use your device less, but still buy one. I think it's difficult.

The shred of hope that I have for decentralized long-form content is tied up in activity pub, I would say. Activity pub for anyone who doesn't know W3C specification for essentially an Open Graph protocol. If you think of it like a web standard for how e-mail works, but applied to a social network like Twitter, so this concept of an inbox to receive messages and an outbox to send messages, where you can build a decentralized network of nodes which have inboxes and outboxes and all of them are able to follow each other and subscribe to one another. I think, that's where things get interesting again, because we start to have long-form content that isn't tied into any one platform, any one network, and that's something I'm interested in exploring.

[1:02:50.3] JM: What's that called? The open pub standard?

[1:02:52.3] JO: Activity pub.

[1:02:53.9] JM: Activity pub standard. I got to do a show on that. That sounds really interesting.

[1:02:57.9] JO: Yeah, you should. If you can get the guy from Mastodon, and that's the biggest social network built on activity pub right now.

[1:03:02.5] JM: I did a show with him. That show was awesome, but we didn't talk about activity pub.

[1:03:06.6] JO: Oh, okay. They started out using a different, I think they moved to activity pub within the last year or so, so maybe they hadn't switched yet. I think that's a really interesting. It's difficult for these things to catch on, because they're so technical. If activity pub is able to be an underlying building block over the web, which most people don't need to understand how it works, they are just able to use it much way, much like they don't necessarily need to understand how e-mail works. They just know that they use it, then there's potential there.

We're certainly looking at experimenting with it in terms of could we connect all Ghost's publications together in a decentralized manner, agnostic for where they're hosted, and allow them the ability to receive each other, send them, receive each other's posts and consume them much like a medium feed, but without needing medium? Yeah, that's something I think I hope is able to have long-form content improve a bit.

I think the second and I keep giving really long answers, I'll just finish maybe with a shorter second part of it, the second part is around aligning economic incentives with the content that people create. I think in in a world driven by ads and ad impressions and the only money that publishers are able to make being from ads, of course what you end up is with is clickbait, of course what you end up with is fake news, because if the only way for a publisher to make money is to get as many ad impressions as possible, to get as many page views as possible, then of course they are going to write the thing no matter what it is, no matter if it's right or wrong, that will get as many ad impressions as possible, and whether that's ten cutest cats on skateboards this week, or some horrific misinformation from a political standpoint that turns people against each other, you're going to end up with that misaligned content because of the economic business model behind it.

I think if we're able to solve some of that as well, if we can get publishers to be able to have a viable business model that is not dependent on ads as their primary source of income, where they are actually serving the readers instead of the advertisers, then we might see higher-quality long-form content. We might see a higher quality journalism as a whole. If your readers are your customers, then maybe you would write stuff that isn't going to mislead them, or disenfranchise them, or piss them off. Maybe you would write stuff for them that's actually useful, because if you don't, then they will cancel, much like every other SaaS business. That's one of the big challenges we're also looking to work on in the next couple of years.

[1:05:34.8] JM: John O'Nolan, thank you for coming on Software Engineering Daily. It's been really fun talking to you.

[1:05:37.8] JO: Thanks so much for having me.

[END OF INTERVIEW]

[1:05:42.0] JM: GoCD is a continuous delivery tool created by ThoughtWorks. It's open source and free to use and GoCD has all the features you need for continuous delivery. Model your deployment pipelines without installing any plugins. Use the value stream map to visualize your end-to-end workflow. If you use Kubernetes, GoCD is a natural fit to add continuous delivery to your project.

With GoCD running on Kubernetes, you define your build workflow and let GoCD provision and scale your infrastructure on the fly. GoCD agents use Kubernetes to scale as needed. Check out gocd.org/sedaily and learn about how you can get started. GoCD was built with the learnings of the ThoughtWorks engineering team, who have talked about building the product in previous episodes of Software Engineering Daily, and it's great to see the continued progress on GoCD with the new Kubernetes integrations.

You can check it out for yourself at gocd.org/sedaily. Thank you so much to ThoughtWorks for being a long-time sponsor of Software Engineering Daily. We're proud to have ThoughtWorks and GoCD as sponsors of the show.

[END]