EPISODE 1059

[INTRODUCTION]

[00:00:00] JM: A content management system or CMS defines how the content of a website is arranged and presented. The most widely used CMS is WordPress, the open source tool that is written in PHP. A large percentage of the web consists of WordPress sites, and WordPress has a huge ecosystem of plugins and templates despite the success of WordPress. The JAMstack represents the future of web development.

JAM stands for JavaScript APIs and markup. In contrast to the monolithic WordPress deployments, a JAMstack site consists of loosely coupled components and there are numerous options for a CMS in this JAMstack world. TinaCMS is one such option. TinaCMS is an acronym for Tina is not a CMS, and it's a toolkit for content management.

Scott Gallant, Jordan Patterson and Nolan Phillips work on TinaCMS and they join the show to explore the topic of content management on the JAMstack.

[SPONSOR MESSAGE]

[00:01:10] JM: If you listen to this show, you are probably a software engineer or a data scientist. If you want to develop skills to build machine learning models, check out Springboard. Springboard is an online education program that gives you hands-on experience with creating and deploying machine learning models into production, and every student who goes through Springboard is paired with a mentor, a machine learning expert who gives that student one-on-one mentorship support over video.

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

you're interested in transitioning into machine learning, go to softwareengineeringdaily.com/springboard.

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

[INTERVIEW CONTINUED]

[00:02:48] JM: Scott, Jordan and Nolan, welcome to the show.

[00:02:52] SG: Hey, thanks for having us, Jeff. We're really happy to be here.

[00:02:56] JM: So, content management systems. For me, the world of content management systems starts with the WordPress just because of how dominant WordPress has been in the consumer ecosystem.

Scott, give me a short history of the evolution of content management systems since WordPress.

[00:03:16] SG: Oh, well. Okay, since WordPress, that's really interesting. I have a certain opinion about CMS as our content management systems, and that I think we've seen little innovation since WordPress. WordPress, you're right. They kind of have dominated the consumer space mostly because it was an open source project that's just proliferated among developers and then it just got used. It's really flexible, of course. It's been used for all sorts of websites.

However, I feel like since the early 2000's when WordPress took off, we've seeing little innovation in terms of like what CMSs can do. One of the most recent interesting innovations is this idea of the headless CMS.

Jeff, I'm sure you've heard of this, but I'm happy to explain it briefly for some of your listeners. A headless CMS just kind of opens up an API on top of your content. You use WordPress to input

API. You fill those form fields and write some markdown – Or not markdown. In this case it would be a WYSIWYG blog post. Save it to the WordPress database. It's now available via API, and your kind of modern frontend can query that API and pull it back.

There a bunch of headless CMSs that take this approach, Contentful being one, Sanity being one, Strappy being one. WordPress and Drupal both to this now, and essentially like every CMS is becoming a headless CMS, where they're being decoupled from the site itself and it's more just talking directly to the content and exposing it via API. I would say like that's a really quick history of CMS, is there are a whole bunch of these things, but this is a big milestone.

[00:04:59] JM: From what I can, tell the motivation for that move towards the headless CMS is largely around the fact that frontend tooling has gotten really, really good and it's becoming easier and easier to build custom website experiences. I think a lot of this is probably due to React just becoming so dominant and such a large and increasingly easy-to-use ecosystem. What's your thesis on the rise of the headless CMS?

[00:05:31] SG: Okay. Well, we built this headless CMS called forestry.io, and it's slightly different that it really interacts with content that's in markdown in your git repository, but let me tell you how we stumbled on that, and I think it's indicative of the greater trend. This is like maybe four or five years ago, Jordan and I – Jordan, who's our CTO and my cofounder and who's on this call. Jordan and I were driving one day and I was telling him that I was building a website for my friend Emily. She ran a restaurant, and Jordan said, "Oh! What are you going to use to build the site?"

At the time I was comfortable using WordPress for sites like this, but I just said, "Oh! I don't want to use WordPress. I don't want to have to deal with like plugging updates or maybe a hacked site in a year from now or two years. This is just a restaurant website. I want to build it statically and I want to throw it an S3 bucket and just forget about it." But I can't, right? Because there was no tooling around this whole static site paradigm at the time. Netlify didn't really exist. Site didn't exist. AWS is becoming a thing for web developers.

The problems that I was touching on was like I don't want to run and manage a LAMP stack server in the cloud, one. Two, I wanted to use a modern frontend framework. At the time I

wanted to use a static site generator like Jekyll or Hugo and I didn't want to build this like PHP thing. I think what's driving this movement is, one, developers want to use modern frontend tooling, and two, we don't want to have a set up like this big monolithic application just to power our websites when we know they can be built statically or server-rendered and delivered really optimally through CDNs to the client.

[00:07:17] JM: The experience of using a headless CMS together or even just your – Actually, I just like to get the spectrum of experiences there. If I am using some decoupled system instead of using a monolithic system like WordPress or like Drupal, what are the frictions there? If I take away the monolithic experience where I can easily manage user roles, and posts, and plug-ins, and all these things that come with the WordPress ecosystem that's made the WordPress ecosystem so sticky, what frictions result from that decoupled ecosystem?

[00:07:58] SG: Okay. Yeah, this is a perfect segue into another project that we announced recently called TinaCMS. I'm of the opinion that the biggest friction point for this new headless paradigm is that the content is so decoupled from the code that the authoring experience is kind of broken. If you use a headless CMS, you're inputting content into four fields, and you click save, and you're like, "All right, you're just kicking me over to the fence and you're hoping that's going to look good." You can't really see the context of what you're doing, whereas the olden days with WordPress, you'd have a preview button and you click preview and it would say, "Okay, here's the stuff that you're working on. This is what it looks like," your blog post or whatever.

Now, things are really decoupled and we often lose that previewing experience, and there are ways to bake that interior CMS now, but it's kind of a clunky experience and sometimes it takes like minutes to build your site in order to preview it and it's just problematic. We've gained all of these developer benefits like choosing the frontend framework that you like. You often React, where we get structured content and we get backed content often. You can deliver your site via CDN, but we lose like this authoring experience, this kind of fundamental thing that's needed for the editing and authoring experience, which is why we built and launched this project called TinaCMS, which is kind of a very different approach to content management and we're convinced it's the future of content management.

[00:09:25] JM: Okay. Well, before we get there, I'd like to talk more about the broader ecosystem and things that have happened since WordPress. We have the evolution of the CMS, but we also have the evolution of the site builder, which is a different class of tool. This is the Wix, Squarespace, Weebly kind of tool. How does usage of those tools compare to usage of CMS?

[00:09:54] SG: This is how we view it, is okay, like the way Weeblys and Wixs and Squarespaces of the world aren't really optimized for the developer experience. We don't bring those into our teams, because they're – It's like Wix has a black box. You don't know what you're getting out the other end. You can't extended it. It's limiting to a developer who knows how to build a system. We don't use those in our teams or we don't put those in front of our clients often, and we use a CMS because what a CMS gives us is the flexibility to control what's under the hood.

With something like open source WordPress, you know you can extend it with plugins. You're not locked-in to this one solution that you don't have any insight into. The tools like Squarespace, Wix, Weebly, WebFlow, etc., these builder solutions, they're all very sophisticated and they're great authoring experiences. A handy person can create great and stunning content using these tools, but developers can't work with those systems. Does that answer question?

[00:11:04] JM: Yeah. Yeah. No. Definitely. The far other end of this spectrum, you do have these kinds of workflows like I create a static site in Jekyll or Hugo and I put it on S3. Make it statically available or I upload it to Netlify, and this is more of the JAMstack style of development. Give me an overview of how the JAMstack has become a common application pattern and how that relates to CMS infrastructure.

[00:11:41] SG: Okay. This is the whole reason why we built Forestry to begin with, was because there wasn't a content management system for maybe JAMstack sites like Jekyll and Hugo like you're suggesting here. This is back in 20 – We kind of launched Forestry in 2016.

The connection there is like, "Okay. Now, as developers, we have this like little application we run on our laptop that compiles markdown files and templates into outputted HTML," which is your website. The question is, is like how do you let other people in in the fun? How do you let a

non-developer edit that kind of stuff? This is what led us to solving this problem with Forestry where we said, “Okay, just point us to your GitHub repo.”

We know what markdown looks. We’ll parse that and build a WordPress style UI for it and we don’t really need to know what the templates are, but if you want a previewing functionality, when you need to know that too, we’re familiar with how these projects are structured like Jekyll and Hugo, Gatsby, NextJS, etc. We can build essentially a WordPress-like UI on top of this new paradigm of static site generators or the JAMstack.

We began doing that. Netlify, CMS was launched soon after we did and then a number of our other people do this too, and we’re essentially applying like traditional content management systems to this new way to organize your code and content for the JAMstack.

[SPONSOR MESSAGE]

[00:13:22] JM: Vettery makes it easier to find a job. If you are listening to this podcast, you are probably serious about software. You are continually learning and updating your skills, which means you are staying competitive in the job market. Vettery is for people like you. Vettery is an online hiring marketplace that connects highly qualified workers with top companies. Workers and companies on the platform are vetted, and this vetting process keeps the whole market high-quality.

Access is exclusive and you can apply to find a job through Vettery by going to vettery.com/sedaily. That’s V-E-T-T-E-R-Y.com/sedaily. Once you are accepted to Vettery, you have access to a modern hiring process. You can set preferences for location, experience level, salary requirements and other parameters so that you only get job opportunities that appeal to you. If you have the right skills, you have access to a better hiring process. You have access to Vettery. So check out vetter.com/sedaily and get $300 signup bonus if you accept a job through Vettery. Vettery is changing the way that people hire and the way that people get hired. Check out vetter.com/sedaily an get a $300 signup bonus if you accept a job through Vettery.

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

[INTERVIEW CONTINUED]

[00:14:57] JM: Netlify, they have some CMS tool. Netlify, for people who don't know, is a hosting site that has more or less become the de facto JAMstack hosting platform. There's also Zeit. I think Zeit has done a good job also. Netlify really has the more popular one. What is the experience of somebody who is using the Netlify CMS approach? What were the, I guess, breakthroughs that Netlify had?

[00:15:29] SG: With Netlify CMS or Netlify the hosting platform?

[00:15:32] JM: Netlify CMS.

[00:15:33] SG: Yeah. Netlify CMS is a pretty brilliant project. It's just like a JavaScript application that's in your browser and interacts with a GitHub API, and they may support other git providers too now, but I think for a long time was initially GitHub. Much like Forestry, you just drop this thing into your site and then you can edit these markdown files and have those edits show these commits on your repo.

It's an open source project. It's like a JavaScript React-based admin for your content that interacts with a GitHub API. One kind of limitation that we found with Netlify CMS that we overcame with Forestry is this ability to have like multiplayer mode. If you're logged in to this CMS in your site and it's just committing to the GitHub APIs and there's no real backend behind any of this, it's possible to trip up and have merge conflicts and that kind of stuff.

With Forestry, there's actually a backend and everything is routed through our backend, and Jordan knows more about this. But we handle – We prevent, do things to prevent merge conflicts. You can have like larger teams editing content that's in your git repo. But yeah, Netlify CMS is like a pretty lightweight open source project in the JAMstack and it's pretty brilliantly architected.

[00:16:50] JM: If we think about using GitHub as a backend for your content management, in contrast to WordPress, WordPress, every WordPress installation I believe by default has a MySQL instance that it gives you. So you're putting the tables of your blog posts into MySQL. If

we're using GitHub as the backend instead of a MySQL database, what implications does that have on the architecture of a CMS?

[00:17:30] JP: Yeah. With Forestry, again, with the static side generators that you've got typically of content files with what's called front matter at the top where meta information is stored. With Forestry, what we do is we interpret those files. We read them and then allow you to create what we call front matter templates that define what the data types are for the front matter.

Instead of the content being stored in a database, it ends up being stored in a flat file in your GitHub repository. Along with a census git, you get automatic version control, the ability to roll back changes. You can see exactly who changed what when and all sorts of things like that.

[00:18:14] JM: Is there anything wrong with just using files instead of using an actual database instance?

[00:18:21] JP: In the case of static sites, I don't think it's an issue, because these are all pre-rendered, right? They're built beforehand. Those static files are taken, compiled into a website and then deploy. They're not read at runtime. They're read at build time. There's no like file system bottleneck that you might expect.

[00:18:42] JM: And that does get at one of the differences between the static site generation workflow and an application like just a naïve WordPress installation where you're not pre-rendering everything. In the static site generation workflow, if I want to have a more dynamically interactive application, like a dynamically interactive CMS where users can interact with it more readily, how does that compare to just a static site generated CMS that I can just run whenever – Like whenever I add a blog post, I can just generate a new version of the static site. If you have actual, like other users that are a multiplayer mode kind of CMS experience, do you have to regenerate the entire site every time some multiplayer, some other user comes in and makes a change to the site?

[00:19:59] JP: Yeah you do, and it is a bit of an issue with most of the static site generators. Hugo is kind of the bit of an exception there because it's so fast to build. But with most of the

JavaScript-based site generators, there's no ability to do incremental builds yet. I think Gatsby is working on that, but you do have to regenerate the entire site. Then as far as deploying, you can deploy it like a differential, but yeah.

[00:20:36] JM: Got it. We've given some context for this landscape. There is a lot of change going on. There's new hosting companies. The react ecosystem is advancing quickly. You've got headless CMS companies, like Contentful. You've got these new static site generation products like Gatsby. You've got a new React-based framework like NextJS. There's a lot of change going on.

I want to start centering the conversation around TinaCMS and Forestry and what you guys are building in this changing ecosystem. Let's start with what TinaCMS is and what it is not. It's not a CMS. It's a toolkit for content management. Why did you call TinaCMS a CMS if it's not a CMS?

[00:21:29] SG: Yeah. So you're referring to the acronym of Tina. Are you familiar what that stands for?

[00:21:35] JM: Oh! I actually don't know what Tina stands for.

[00:21:38] SG: Tina is a recursive acronym. Do you know what that means, Jeff?

[00:21:41] JM: Yeah. Yeah.

[00:21:41] SG: Yeah, like GNU, not Unix.

[00:21:45] JM: Oh! Tina is not a TMS. I get it. Okay. All right. I get it. T-I-N-A.

[00:21:50] SG: Oh! That was a test. You just passed. I think Nolan might be the best person to take this one away.

[00:21:57] NP: Yeah. As we kind of mentioned, there's sort of a few different stages of content management where we start up of WordPress where the entire website was built on your CMS. The display of the content and the content management were all put together in one. Then you

have the headless CMS where it's still this running application that you used to edit your content, but that part of it is separated from your website. You're editing your content using a different application than what you're building your website with.

We say that Tina is not a CMS because it's not really either of those first two things. It looks a little bit on the outside more like WordPress where you're editing experiences baked into your site, but the key difference is, with WordPress, you're building your website on top of WordPress. With Tina, you're building the CMS into your website.

We provide a set of tools that make it easy for you to build those editing interfaces directly into your website so that it's not really a separate application. Your website itself becomes a content management system.

[00:23:18] JM: This is pretty cool, because if I think about most blogs on a company, so like I go to – Like I did an interview recently with a company called LogRocket, and they have a link to the blog in the header. I click on the blog and it takes me to a part of the site that probably is a part of the site, but it looks like it's kind of different. It looks like it's maybe they're using Ghost, which is a blogging tool or it could be a WordPress thing. I think it's Ghost or one of these other kind of next-generation blogging systems. But nonetheless, it does not feel heavily integrated with the rest of the site. It feels like this different part of the website, which is okay, because like I'm just reading blog content, right? Why does it matter if this blog is essentially disconnected from the rest of the site?

[00:24:17] NP: It's not just that the blog is disconnected. It's that the editing experience for the blog is disconnected. If I want to create a new blog post for release notes on TinaCMS, then I don't go to some separate application. I am actually going to tinacms.org/blog, clicking the edit button and now I'm looking at the blog post as it will look when it's published to when it's actually published. I'm not seeing a representation of what my blog will look like. It's really a WYSIWYG, like what you see is what you get, not just what you see is sort of like what you're going to get.

[00:25:00] SG: Jeff, just to use an example. On the Software Engineering Daily website, I saw that you have podcasts and you have blogs in different categories of content. When it comes time for you to add this show to your website, say, you had Tina installed. You would just go to

your public website, go to bot podcasts, click login and then add new podcast and you're basically looking at an empty page on your website where you start filling out the title running either your show notes, etc. etc. right on your page on not in a separate admin.

This is all built on top of like a developer friendly world of using modern frontend tooling on the JAMstack. We think of it as building like a world-class editing experience on top of a world-class development framework development or development experience.

[00:25:53] JM: Right. If I think about the experience of using Medium to create a blog, if I'm writing my blog post in Medium, the blog post ends up looking very much like it appears in my editor, whereas in WordPress, if I am writing a blog post, I'm doing in this editing experience. It's kind of a cramped experience.

I do most of my creative work in Google Docs and then I copy-paste it to WordPress, which is not a wonderful experience, and ideally it would be more of like a WYSIWYG kind of experience, like what you get – This is what you get out of like Wix, or Weebly, or Squarespace where you can kind of see more intimately what your content is going to look like as you're typing it.

Now, that seems kind of tricky, because you have to essentially embed a content editor along with the other live components of the website. What are the difficult engineering problems in making that happen?

[00:27:01] SG: It's a pretty challenging problem. There're kind of two aspects of it, which is how do I actually make that interface editable and then how do we actually save this data? The first side is with React components, and we spent a lot of time building out React components and APIs that make it easy to embed content in your page and make that content editable.

Our intention is always to make it so there's some default styling along with that. So you can jump in and get it going really fast. But if you need to, you can kind of extend that and modify the styles to best suit your website.

The other side of it is figuring out how do you actually save that data somewhere. It needs to get saved, and that's probably one of the bigger challenges that we've had. One approach that we're pretty excited about right now is NextJS 9.3 released a preview functionality. With NextJS, it's kind of a hybrid framework. Whenever you deploy NextJS, you can deploy both a static version of that site and a dynamic version.

The static version is all prebuilt. All the content is rendered to HTML and it's served up statically on a CDN as you – If you go to our website, you'll see this big edit site button, and when you click that, it switches that around. You switch into preview mode, which essentially means instead of just hitting the static cache, let's actually hit the API roads now. When you do that, it makes it possible to add a little more dynamic behavior there. You can do authentication through a server request. You can set up a web socket connection to some other thing. We haven't fully explored what we can do with that, but this approach, this hybrid approach of having that static output but also being able to fall back to a dynamic one to introduce some of that more dynamic behavior that you need for content management system is looking pretty promising.

[00:29:14] JM: So if I want to turn my WordPress website into a more dynamically editable website, let's talk about what a workflow might look like. I've already got this WordPress installation and it's monolithic, it's not a pleasure to work with, but it has the benefits of being monolithic and all there and it works and I can create new roles. It's actually like it's pretty good. It's not perfect, but it's not great. That's the story of WordPress. But if I wanted to have a completely clean approach and I wanted to move this entire content management system that's monolithic in WordPress towards this place where if we have the Software Engineering Daily post about this episode, I can just click on component at a text paragraph in that post and edit it and have a live editing experience. That's obviously not going to be possible with the WordPress thing today. If I wanted to actually migrate it, what would be my steps for doing that? If I want to migrate to some other CMS platform, a more configurable, more non-monolithic CMS platform, what are the steps for doing that?

[00:30:39] NP: I think there's kind of maybe two or three steps. Step one would be to stop using WordPress to generate your website. You could still use of a content management, but you instead use Gatsby or Next or GridSim, any of those other static side generators to pull the content from your WordPress site and render it using one of these new tools.

If you are looking to have that really like on-page editing experience, then step two would be like use TinaCMS to put that editing capability into your website and set up the API so that when you hit save in those forms, it saves back to WordPress. In this case, this part of why Tina is not a CMS, because you can use it with other CMSs. You could have TinaCMS powering the editing experience on that frontend giving you that awesome kind Squarespace WIX style editing experience but still be using the modern frontend frameworks to build your website. If you're still on WordPress, then you don't need to totally ditch it right away.

[00:31:57] JM: The new website that I would be building with using WordPress as my backend, I would have to build it in React components, I believe, and use Gatsby as my way of doing pre-rendering of that site, of generating the statically generated version of my WordPress site. Gatsby can connect to my WordPress backend and create all of the components that I need to have a full website experience. Then if I wanted to add editability to the statically generated website, I could use TinaCMS or I could use Forestry?

[00:32:43] NP: Tina and Forestry are fairly different in the way they work. Forestry would be more of a straight replacement for WordPress. You would be switching from WordPress to using Forestry and files in git as the storage medium.

[00:33:01] JM: Got it. Let’s actually talk a little bit about the story there. You guys started working, or I guess Nolan started with TinaCMS, and at some point, Scott and Jordan, you both started working with Nolan, and then eventually TinaCMS and your work there evolved or was augmented by your work with Forestry, which is a company that you’ve actually built. TinaCMS is the open source project, and Forestry is a company that you guys have built. Can you give me the timeline for these different projects?

[00:33:36] SG: Yeah. Like I said earlier, we launched Forestry in late 2016, and Forestry is more of like Nolan was just saying, it’s a CMS that is a CMS. It stores your content. It's an editing admin that writes to that content. That content happens to need to be stored in git. When that's compared to something like, say, Contentful, or Sanity, or WordPress, those systems are [inaudible 00:34:06] database. Forestry commits it to your git repository.

We launched in 2016 and we built this whole thing. We watched the JAMstack, kind of this whole space evolve and blossom. Like I was mentioning earlier, one big problem we noticed is this disconnect between editing your content and writing it in the CMS admin and previewing your content, because things had become so decoupled. We always had this vision of where we're getting to now with Tina. This vision of content editing should be much more like editing a Google Doc instead of editing like a WordPress site.

Now think of your own case, Jeff. I found it really interesting to say you write a lot of your notes or your articles in Google Docs and copy and paste them over to WordPress and that just tells me that like the CMS software you're using isn't good enough, because you have to use another software to write it and then you just copy and paste it and drop it into this thing.

We've always had this philosophy internally in the company that our vision for content management is much more like a Google docs and it gets this like a really great authoring experience on top of the proper development stack. We do the hackathon at Forestry in 2019 where we said, "Hey, would this be possible if we just focus on like, say, the modern React frameworks now, like Next and Gatsby and written in just React?"

At the end of this Hackathon we're like, "Hey, this is actually possible now." We should get this open source project at the door, and like this is really in line with our vision. Let's launch this thing," which we launched at the JAMstack Conference in October 2019 in San Francisco "Let's get this thing out the door and built some community around it and get some momentum around this whole idea, because it's quite a new paradigm." Eventually these roadmaps merged, where this live editing stuff could be opened up to all the Forestry users. But for now, we just kind of blaze ahead and launched it as its own open source project.

[00:36:08] JM: Tina CMS came after Forestry.

[00:36:12] SG: A few years after. Correct.

[00:36:14] JM: Okay. Interesting. From a company perspective, what's the value in creating an open source content editing, content management system?

[00:36:27] SG: Yeah. Before I answer that question, maybe I'll tell you that the mission for company is we want to help people build a better web, and we don't say developers and we don't say content people. We just say people, because we feel like that's like sometimes we build developer tooling and sometimes we build kind of these editing experiences, but it's like the workers of the web. We want to give them really good tools so we can strengthen the web, and people can put out really good content and help educate the world.

We feel from like the mission statement from the organization, it's a no-brainer to build an open source project. We really believe in open source. Can this be monetized in the future? Well, in one way it's monetized now where it just generates attention and people learn about our company and find out about Forestry and pay us for Forestry. I would say it's more or less in line with the mission statement than the business.

[SPONSOR MESSAGE]

[00:37:31] JM: Over the last few months, I've started hearing about Retool. Every business needs internal tools, but if we're being honest, I don't know of many engineers who really enjoy building internal tools. It can be hard to get engineering resources to build back-office applications and it's definitely hard to get engineers excited about maintaining those back-office applications. Companies like a Doordash, and Brex, and Amazon use Retool to build custom internal tools faster.

The idea is that internal tools mostly look the same. They're made out of tables, and dropdowns, and buttons, and text inputs. Retool gives you a drag-and-drop interface so engineers can build these internal UIs in hours, not days, and they can spend more time building features that customers will see. Retool connects to any database and API. For example, if you are pulling data from Postgres, you just write a SQL query. You drag a table on to the canvas.

If you want to try out Retool, you can go to retool.com/sedaily. That's R-E-T-O-O-L.com/sedaily, and you can even host Retool on-premise if you want to keep it ultra-secure. I've heard a lot of good things about Retool from engineers who I respect. So check it out at retool.com/sedaily.

[INTERVIEW CONTINUED]

[00:39:07] JM: So let's talk more about the open content editing experience. If I generate my static website using Gatsby, I take my WordPress blog, I have Gatsby replicated as this new nifty Gatsby site that maybe I've made all these React components and then now it's just built purely from React components and it's much faster, it's much cleaner, it's all editable because I have built it myself from React components. What is the process for using TinaCMS?

If I use TinaCMS and I make open content editing. I make it possible for different people to edit the generated static website. What is TinaCMS doing on my frontend? What's the runtime experience in the browser and also how durable is it? If I'm using Google Docs, for example, it's like auto saving all the time. I want to understand how that compares to my editing experience for this Gatsby editing site.

[00:40:18] SG: Yeah. This is a question that we're really in the throes of right now, and kind of the annoying answer is it's sort of depends. As I mentioned earlier, you might have Tina talking to a WordPress backend. Well, Tina is sort of just the frontend, the interface builder side of it, and it depends a little bit on like what you're using as a backend that determines some of the details here. It's hard to come up with a generic answer.

Again, Tina is not really a standalone application that's running and doing this. It's a framework for helping you build this. If you're using Gatsby and the file system, then in that case you will have kind of a staging server running. There're a number of people using Gatsby cloud to do this. Some people have spun up their own staging servers, and Gatsby is running in development mode. As you're typing and you're making changes to your blog posts or your homepage or whatever, Tina ends up writing back to the file system. By writing back to the file system, that will cause Gatsby to rebuild the site and update the previews that you're looking at.

What we're working on now is a slightly different approach, where you have a NextJS side that is using this preview functionality we were talking about. With that functionality, there isn't really a file system running. Instead of loading the content from the file system to edit it, we fetch the content from the GitHub API. Any kind of intermediary changes are just rendered on the page immediately and kept in local storage to make sure that you don't lose them if you refresh the

page. Then when you go to hit save, that will make a commit through the GitHub API to persist that change for everybody.

[00:42:22] JM: That's pretty good. This is one thing some people may not know about browsers. I didn't know it until maybe a year ago, that – And this is pretty naïve, but if you're using a browser, your browser has access to disk space. Your browser can write durable data to disk. Just like you're using Microsoft Word on your computer and Microsoft word is saving to disk. Your browser can access disks storage. If you were editing something in this application description that you just gave, then you can snapshot to disk, if my browser crashes, you can have that data save to disk and you can refresh it or present it back to the user when the user opens back up the browser and navigates to the site. That of course doesn't protect your computer from spontaneously combusting, which is why we prefer to be saving to the cloud. We prefer the Google Docs connected kind of experience. But if you're using GitHub as a backend store, I'm not sure that the GitHub API would let you save as aggressively as you might want to. How flexible is the GitHub API there? Can you use it for this kind of like a really aggressive backup?

[00:43:46] SG: I'm not really sure that it would be appropriate for really aggressive backup. I'm talking backing up every second or something. I haven't spent a whole lot of time thinking about this problem in the full detail, but we know that there could be solutions for this down the road. If you're using some other service maybe that keeps track of those kind of unsaved changes so you don't need to be committing back to the API all the time.

Yeah, that's definitely a very difficult problem and one that is on our mind. It's one that we did find a solution for Forestry. We have some ideas, but at the moment you're just using local storage as your storage mechanism. Again, if you're using back to that Gatsby example where it's writing to the file system, well, it's writing to the file system. Any uncommitted changes are in your file system as just the uncommitted changes.

[00:44:54] JP: Just to expand on that a bit further, we have plans for – Or what we've been calling internally Tina Teams, which will provide like services like this. We can act as an intermediary between the frontend of Tina and whatever API or saving content to and provide that kind of durability in between.

[00:45:17] JM: Cool! How do you expect this changing the market, or how do you expect the market to change in the near future to your advantage? If I think about the online publishing experience today, it is very much this kind of like I give a user access to the WordPress publishing process and that user can then login. They can publish a post. They can make a new article. What predictions do you have about the way that the market is going to shift in the direction of people wanting a more open content editing experience?

[00:46:00] SG: Yeah. It doesn't necessarily need to be open. If you're referring to – We published an article recently about open authoring where if you want anybody to edit, say, your documentation, say, you can add Tina to your documentation site and integrate it with the GitHub API. Now say Joe from Texas can just go to your website and click edit and edit it kind of like a Google Doc and open a pull request right there not from GitHub but from your live site. That's open authoring, and Tina isn't just for open authoring. It's for teams working on, say, internal sites or public marketing sites where they may not want the whole world accessing.

But your question, where do we see like the industry or the content editing needs going? When we first put Tina on our website like to dog food it and test it internally and stuff, what we noticed was people – We are all creating content way more frequently because the friction is so low. The barrier to editing content on our site is we just visit it and click add new blog post and start writing it right there, and we're doing it – In our case, we use git as the storage medium. So you're doing it on your own branch. What we found is just like, "Okay, content –" As soon as we made this available to our own team, content contributions went up like crazy, because now everybody is also running blog posts and it's not this kind of convoluted process of you write it in Google Docs and you copy and paste into WordPress and you try to like send somebody a link. It's much more fluid and more of a developer workflow. We think this – How this will impact web teams is that the friction will be so low that people would just be able to create more great content more easily on top of a really modern developer stack.

[00:47:54] JM: Cool! Tell me more about the business of Forestry. What are the products that you have today and what are you hoping to build?

[00:48:04] SG: Right. Okay. Forestry is the pretty traditional content management system. It's close source and it talks the your git repo. Like I said before, you point it at your project and say, “Hey, my Jekyll site or my Gatsby site or my NextJS site is over in this repo. Make an editable admin for me. Then Forestry goes ahead and does that, and now you can invite people. If you go to our website, you'll see that there’re some fairly big organizations using Forestry. A few of them are listed there, Mastercard, and Spotify, and Sketch and there are others too, and these people need much more than that. They have like security requirements and they have maybe teams of users of the manage and just like bigger needs. With those things, they end up being in a premium planet Forestry, and that’s essentially how th business works. We have a free plan. If you're off the top of your show, I talked about an early version of myself building a website for my friend Emily who owned a restaurant. That kind of use case really belongs in a free offering, or something like Netlify CMS, which is open source and free. But if you’re, say, a larger organization, you have many users and teams of users even and many sites, you'd end up in a premium package for Forestry.

[00:49:24] JM: That's pretty nice onboarding experience, because I can imagine there's a lot of people who have sites that they've built and then they just add a very rigid Jekyll or Hugo-based set of posts, and then if they want to have a more actual CMS like experience at a WordPress like CMS-like experience, a WordPress-like, CMS-like experience on top of tha, then it’s a pretty nice roadmap to getting to that CMS-like experience.

Just to close off, I’d like to get your perspective about the market, like how does the CMS market and the JAMstack evolve over the next couple years? Do you have any predictions for the broader market development?

[00:50:16] SG: Yeah. We’re of the opinion that every website is becoming a JAMstack site, every website that a developer is touching. If you were to rebuild Software Engineering Daily today, what would you use? I'm just curious, Jeff.

[00:50:32] JM: Well, we actually did rebuild. We have this thing called softwaredaily.com, which is actually like somebody in our community built a platform for us. It’s a VueJS website, and we have mobile apps. But it's a VueJS, NodeJS, but this was started like three years ago. It was

kind of pre-JAMstack becoming the standard. If I was to completely rebuild today, it would be React. It would be probably using NextJS. Yeah, that's what I would build it off of.

[00:51:07] SG: Yeah, and every other developer in the planet is kind of thinking around this, thinking similarly about building websites, where we kind of want to use JavaScript in the frontend. We no longer want to build like PHP templates like we used to do one day with WordPress and we really care about like the way that frontend code is optimized. We want to serve from CDNs like we can, like have it pre-rendered in servers and serve from CDN.

Yeah, we're just of the opinion that every developer is thinking this way and we still want our structured content. So we want the content to be stored somewhere else where it makes sense. Yeah, we want the benefits of the modern stack. So that's kind of where we see the future going. Obviously, it's not going to be every developer in the planet, but we just think the majority of sites are going in this direction, which is why you're seeing projects like Netlify, Zeit, Takeoff, you see the frameworks – Like we mentioned a bunch, but Gatsby, NextJS, GridSum, Hugo, Jekyll, like these things that will takeoff. What you're saying, headless CMS becoming a real thing, it's because, okay, we need this decoupled nature now.

Yeah, in terms of like market, that's where we see things going, is just proceeding in this direction of, "Okay, people are going from monolithic applications to the JAMstack, and a big driver is just JavaScript being the language of choice for web developers.

[00:52:39] JM: Awesome! Well, thank you guys very much for joining the show. It's been a great conversation and I have enjoyed talking to you.

[00:52:44] SG: Okay! Thank you so much for having us, Jeff.

[00:52:46] NP: Yeah, thank you.

[00:52:47] JP: Thanks.

[END OF INTERVIEW]

[00:52:57] JM: You probably do not enjoy searching for a job. Engineers don't like sacrificing their time to do phone screens, and we don't like doing whiteboard problems and working on tedious take home projects. Everyone knows the software hiring process is not perfect. But what's the alternative? Triplebyte is the alternative.

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

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

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

Thank you to Triplebyte.

[END]