EPISODE 1386

[INTRODUCTION]

[00:00:00] KP: When creating a website, there's no shortage of choices for how to do it. Builders must make strategic decisions about the language or framework they want to adopt. An important first consideration for many is selecting a web application framework like React or Vue. Motivated by a low page response time and a good user experience, many developers want their site to be server-side rendered.

Nuxt.js is a free and open source web application framework based on Vue.js, which, amongst other benefits, brings server-side rendering to the Vue.js developer. In this episode I interview, Alex Lichter, founder of Developmint and Nuxt.js maintainer. We discuss the features of Nuxt and what role it can play in your next web application.

[INTERVIEW]

[00:00:48] KP: Alex, welcome to Software Engineering Daily.

[00:00:49] AL: Thanks for having me. Hey!

[00:00:51] KP: Where did you get started with software?

[00:00:54] AL: So I think it all started back in school actually when I was like 13 and I've had – Like, yeah, I used computers every now. And then and it was quite fun. But then I felt like, "Okay, I can do more than just using it. I want to like take a little peek inside how things work." And I found out like writing small programs wasn't that tricky at first. And then I started exploring. And I found out it's a bit trickier than I thought. But still I loved it.

[00:01:18] KP: What about first languages? Where were your early successes?

[00:01:21] AL: Ah, yeah. Well, that's a tough one. So my first language was Java actually for quite some time, because I started with like the whole Minecraft type. And I love to play games.

So Minecraft was like the hype back then. And there were like multiplayer servers and you could like write plugins and mods with Java. So I started learning it. And yeah, my code was super crappy, but it was so – It made everything so cool.

So yeah, I learned Java and I kept on going. But then I found out like PHP is a bit more fun, like less boiler plate, getting things like done quicker, and also like having the web as a platform felt so amazing to me. Yeah, so that was PHP. And finally I still use PHP every now and then, but now I'm mainly using JavaScript and TypeScript as languages.

[00:02:07] KP: Well, JavaScript, and I guess TypeScript as well, have been largely grown especially in the web space around frameworks. What were some of your first exposures to different frameworks?

[00:02:18] AL: So, yeah. Well, the first frame I actually learned was Laravel in the PHP world. And for those of you who don't know, like the Laravel founder, Taylor Otwell, and the Vue.js creator, Evan You, they're kind of friends I would say. Also there's like some community friendship between the two frameworks. So it was kind of natural. Like Laravel came with like a Vue template or like a Vue starter kit. So I tried it out. And that's where I like got like pulled into the whole Vue universe. It felt like – Before I used, of course, like JQuery and stuff also when like browsers hadn't their own standards very well. But that's settled down, I mean, now we have ECMAscript and stuff. And I felt like Vue was such an amazing way to like organize code, make things reusable. And I started using it also like not only sprinkled into a PHP application, but finally also as the whole front-end solution.

[00:03:06] KP: So I'm sure you're familiar with other frameworks. React and Angular are two that come to mind. What makes Vue more attractive, or at least for you?

[00:03:15] AL: So, yeah, of course, I use both of them not on a daily basis. So for me, like Vue was easier to grasp when I had like JavaScript HTML CSS knowledge, because basically have something called a single file component. And in there you can add like a template that is basically HTML with a bit of Vue magic added. So you have some kind of directive where can say render this part of HTML, or loop through this in this data. Then you have like a script tag where you can add like your business logic, so JavaScript. And you also have a style tag for that

component where you can add like component created styles. And this is something that I kind of missed at the other frameworks to get like this entrance barrier was quite high. Especially for Angular, it's like very heavy. And you have to learn several concepts. And also TypeScript, which I didn't learn before. I started with JavaScript, as I think many people.

And with React like I felt it's as mighty as Vue I would say. Also, it can be relatively lightweight. But the good thing with Vue is that you don't have to like track your reactivity manually then like say when the component should rerun. That Vue is doing that all under the hood with your own like reactivity system. And that's really appealing because you don't have to deal with that at all. You just say, "Okay, this is reactive." And whenever something based on this, say, data property changes, then Vue is just doing it automatically under the hood.

[00:04:35] KP: And what is Nuxt.js?

[00:04:38] AL: Yeah. So Nuxt is a framework based on Vue.js. And across like my journey in the software development world, I came across Vue and used it a bit. And when I used it as a framework for like the whole frontend and used PHP Laravel only like as an API as a backend, I realized that Vue is amazing with its component system, with its ecosystem, with the community, which I really love. But at some point, just using Vue, it had some issues, especially when it comes to SEO. Because Vue is a single page application framework. And that means that the JavaScript is generating the HTML. So when you open a page, you don't have like the HTML inside, but just an almost empty page with like a loading spinner. And then the browser executes the JavaScript, and eventually renders the HTML. But that's not good for crawlers.

And I worked on a few projects that were like public-facing so SEO was much needed. And then, well, Nuxt was more or less the optimal choice. There was even recommended in the Vue guides, because, as I said, it's a framework on top of you and it does something called server-side rendering, which basically means that it does everything Vue does on the browser side. It does that already on the server-side and then sends into HTML. And then Vue takes over and it's normal SPA again.

And yeah, besides service-side rendering, Nuxt has lots of other benefits. Like it has automatic routing. So, as I said, I come from the PHP world. And you add an about.php file and you can

just open /about and it just works. And that kind of simplicity was always fascinating. And with Nuxt, this is possible again. Like in Vue, you have to write a file called router.js, which says, "Okay, this is your route. This maps through this in this component. And these are like the placeholders and parameters." And Nuxt utilizes the file system to more or less build this router.js file based on your files you have in your folder. So there are lots of cool features in there. And yeah, I'm using this for almost all projects wherever I can because I feel like it's really powerful. It's easy to get in especially because it's based on Vue. You can write prototypes super fast. It's enterprise-ready. So basically available for almost all kinds of web projects.

[00:06:48] KP: Well, many people can say, "Oh, yes, it's enterprise-ready and there's some benchmark." But there's a pretty good list of stuff in your gallery. Do you care to highlight a few of the major projects that are using Nuxt?

[00:06:59] AL: So, yeah, there are lots of companies. So Nuxt has a few partners. Like we have Storyblocks, Trappy, for example these are I would say like the common headless CMSs nowadays. But for example, Microsoft is using Nuxt.js for its PWA builder. So there's like a side where you can set up like a service worker for PWA. I think Adobe is using Nuxt as well. We have even GitHub using Nuxt, for example, for the GitHub Stars website. So there are lots of larger companies that try it out. Even NASA used it for their website for the Mouse Rover. So lots of amazing projects. And there's also a list. Maybe we can link that in the show notes that like highlight sites that are built with Vue and also with Nuxt. So that could be quite interesting to see some examples from like larger and also smaller sites that dabbled with it.

[00:07:46] KP: So I know very large organizations, and for good reason, are hesitant to adopt what they perceive as new or risky technology. They need to have long lasting stable things. Are there any lessons learned in the journey of how to make something that's attractive to enterprise groups for adoption?

[00:08:05] AL: So, yeah, especially as like working in open source. There's always something you have to keep in mind. If you add a feature, especially from like external contributors, you always have to think about that you have a cost of maintaining. So it might be a super cool feature. But once added, you can't easily undo it. Because there are people like relying on it. There are people maybe building something upon it and so on. So at first it's super important to

think about can we or should we add this to the core? Or could it be like a module? And I think that's also a super important part.

Something that should be enterprise-ready is ideally modular, because that means people can like shape the framework as they need it. Not only by like just adding things, but also hooking into the framework, into the process, and alter some functionalities as they like, as the requirements say, for example. So being flexible while not adding that much to the core and keep the core lightweight I think is a quite good strategy here. Because then you can like add more modules. You can share them. You can test them in isolation. You can reuse them. I think that's one of the main lessons learned here.

[00:09:14] KP: So there's – I'm not sure if it's a React framework. I haven't used it myself, but there's a system called Next.js. Is there any reference there or is this just a collision of a very close spelling?

[00:09:26] AL: So, of course, there are references. So back then, when both Sebastian and Alex Chopin, the original authors of Nuxt, when they came up with the idea, it was quite close after Next.js' release. So they thought, "Okay, Next plus Vue." So the U from Vue, Nuxt, there we go. But besides the name. Of course, like the target and the idea is kind of similar based on the ecosystem. So of course, Next.js is a framework based on React. Nuxt.js is a framework based on Vue. But of course, the approach totally depends on the ecosystem. So it's not that like Nuxt is just a Next.js copy. And also the directions are a bit different depending on what kind of features you're looking at.

But the most important part is, of course, there is always inspiration especially open source. Like we have so many frameworks, so many static site generators, so many new projects. And if there is a good idea like, for example, what would Gatsby and Turbolinks did with preloading JavaScript when you like hover over a link to another site or when the site comes into the viewport so it linked to the site. These ideas are adapted quite quick through each framework, because if that fits in the ecosystem mindset, they will be added because it brings benefit to the developers. Yeah, I would say that's a quite good explanation regarding the connection between Next and Nuxt of course.

[00:10:43] KP: Yeah, definitely. Could we talk a little bit about common patterns for data access? Let's say I've got that API. Maybe it's in Laravel like you were describing, or some other language, and I'm just going to connect via REST. How easy is that to do in Nuxt?

[00:10:57] AL: So that's a good thing. Again, Nuxt is quite flexible. We have a couple of modules. So you can, for example, pull in Axios. You can also just use the browser's fetch API, which you also probably fill on the server so there's no problem on the server-side. If you have like a GraphQL API, you can also use like Apollo or any other GraphQL client if you want some. So that's the nice part. You're completely free to choose whatever data fetching library you want. And we don't force you in any direction. Of course, we have modules ready to give you a good head start if you say like, "Okay, what is the thing that's recommended, or it's battle proven, or that's just common but people use?" And you can choose the module and you're good to go.

But there is nothing like – We don't have a GraphQL layer or something in place. The only thing we have is like inside your Vue components and/or inside your page components, so the components that are used to render the page, but also any other Vue component, you can specify through a method called fetch what kind of data you want to pull in and want to use in the components and maybe pass further down. So yeah, lots of flexibility. But there are also some like common patterns like adding a repository pattern so you're not really dependent on what data fetcher library you use when using your API.

So say in a Vue component you just say dollar api.gets, or .users.gets and then maybe an ID or just getting all users or all posts or whatever. And then of course you build inside the repository the whole access. So at some point when you say, "Hmm, maybe my data fetching library, we should switch that." You don't have to like change all the code things, but just change it into repository.

[00:12:31] KP: Do I need to know Vue as a prerequisite? Or can I come directly to Nuxt?

[00:12:35] AL: That's a good one. So I think you can start straight with Nuxt, because you can just write Vue at the beginning. You don't have to know lots of like Nuxt internal stuff. And you can learn that step by step. But of course if you know Vue in advance, and of course if you have like a basic understanding of how reactivity works, that would really help.

Anyway, you can just start and say like I read the Nuxt docs. I set up a Nuxt project. And then I take a look onto the Vue documentation and do what they do with single file components in my Nuxt project, and it will work the same way.

[00:13:06] KP: What's the story, if anything, for state management?

[00:13:09] AL: Yeah, so state management. Generally, in Vue, we had Vuex for quite a long time as like the main state management solution. With Vue 3 and the Composition API, there are a couple more patterns. And there's also a good talk which I can recommend here by Vanessa Otto, who talks about state management with like Vuex with so-called stateful composables. So you use like the Composition API and have some global variables more or less in the composables to keep your state. And also about, for example, Pinia, which is another state management library by Eduardo, who also shows like how fresh pattern could look like with the Composition API.

So TLDRs Vuex is the main solution. But you can also, as I said, use Pinia, you stateful components. Also use XState, for example, if you want like a whole state machine, which also is really powerful. Again, you're not bound to anything. You can use what's there and what's battle tested. But you can also try out new things.

[00:14:07] KP: I really like the feature you were describing earlier that was sort of PHP inspired, that I don't have to create a router and figure out routing. Just put a new file in a new directory and it should appear there. Could you talk a little bit about what it takes behind the scenes to make that work?

[00:14:22] AL: Absolutely. So to make the work, we have a folder called pages, which is option on Nuxt 3 in the beta released a few days ago. But Next 2, it's mandatory. And in there, you add the files. So you add an about a Vue for your component there and all good. Behind the scenes, what's happening during the build step of the Nuxt application is that Nuxt looks through this pages folder. And based on the folder structure and the naming of the files, it will understand, "Okay, this is just a static site about a Vue." And maybe this is a folder. So /slash users, for

example, is a folder. And in there we have something called, say, _slack the view. And thanks to this underscore, Nuxt knows, "Okay, that's a side that takes a parameter and it's called slack."

So based on the file structure, Nuxt builds behind the scenes router.js and maps it. And you can even see like the generated router.js in the so called .next folder which contains like all the generated files. So yeah, basically Nuxt using the file system to just do it for you and you don't have to figure it out and to write it on your own.

[00:15:26] KP: What if I want to have something a little dynamic? Like we started out the conversation at SEO, and I do want to get back to that. Perhaps I want some sort of slug version of my title in the URL, but I don't want to create a page for every post. How can Nuxt help me?

[00:15:40] AL: Yeah, we have this – As I mentioned right away, this _slack the view. And in there you can like fetch the data based on the parameters. So you can use the parameters that, say, like you call post/my-awesome-post, and this name, this post then can be used inside this _slack component. And based on that, you can fetch like SEO data. And generally, we also post data from CMs, be it a headless CMs, or be it a good old WordPress with an API, or your custom whatever API, or just something to test it out. That all will work.

[00:16:13] KP: Makes sense. Well, yeah, having a page – And I had this experience too the first time I started dabbling in single page apps. You create this cool thing. But if you just crawl that, it's a script tag essentially, to a lot of crawlers. I can see where that's true broadly. Is that true of something like the Google and the Bing crawlers, if those are the only two I care about?

[00:16:34] AL: Yeah. It's not easy to answer, because those two can understand JavaScript nowadays. But that also means that they can potentially index your site. Still, it doesn't mean that your site is fast, because they still have to wait until the HTML is being rendered by the JavaScript. So it still is useful to use something like server-side rendering to like reduce the time to interactive. To show to crawl the whole HTML to make it a bit more resilient. Because if the JavaScript fails, you get just a white blank page and so on.

So I summarized it in my talk about SEO and Vue.js world where I said like you can use just a plain Vue single file application if you – Or a single page application if you like that. But if you really struggle especially like in a highly competitive space, you should use server-side rendering or a static site generation to actually like level-up your SEO game and improve also the core web vitals, the site performance. This all goes hand in hand.

[00:17:28] KP: do you have any rough metrics around that? Obviously, it's going to depend a lot on implementation. But do people see a 5X? Or what's the speed up?

[00:17:36] AL: So as I said, that's tough to tell. I don't have definite numbers, because it highly varies. The main point is that you move lots of calls. So the initial request will always go to the server if you use like dynamic server-side rendering, say, on an ecommerce shop. And the good part here is that you can do lots of caching on the server-side, of course. Ideally, that API call maybe to get your product is already cached because somebody else asked for the product in the last minute or in the last hour, whatsoever, depending on how you handle your data.

And then the time. So the time to first byte is super quick. So the time from the server outputting HTML is ideally not or almost not influenced by that API call because it's cached and you have the opportunity to cache it in there. And then you get the HTML. And then there's a process called hydration, and Vue will take over and you can also optimize that a bit by saying like only hydrate the things you can see. So if you will like take the HTML generated by the server and then transform it into its own like virtual DOM representation so it can apply events, make things reactive and so on. If you wouldn't apply like caching and stuff, then you mainly move the metrics from time to interactive to like time to first byte. So you reduce the time to active, but you increase a little bit the time to first byte because an API request on the server-side has to be processed. We have to wait for this. So otherwise data wouldn't be there.

The good part is there are lots of opportunities to optimize. There is, as I said like, caching. You could cache the API call. You could cache the whole response. And I think that's what the main power is here. For static site generation, it's a whole other game, because you don't generate the HTML on the fly all the time, but just want some build time. And then you have like almost like zero time the first byte due to any API calls because they already happened. And I think that's a real game changer here.

So the whole Jamstack part that also Nuxt can cover, because then you don't have any API calls and you just get the HTML out of the box. It's just lying around on any CDN or wherever you host your files. And that that will lead to a huge speed up. But unfortunately, numbers are tough to provide. It depends a lot on the application. But I guess there are lots of Jamstack statistics when like larger companies switched over the site and they said like 20X, I think, speed up if I remember correctly. So yeah, I think also checking out Netlify's ebook about Jamstack. They released one I think half a year or a year ago that has some compelling numbers in there.

[00:20:03] KP: Are there any ways in which developers can extend or integrate software or do plug-ins and things like that for Nuxt?

[00:20:10] AL: Absolutely. Yeah. So we have a full-fledged module system. So you can like add, I would say, almost anything you like. That means like adding new page components, changing configuration settings. And also adding Nuxt plugins so you can do that inside your application. But you can also do that in a module. And Nuxt plugins are more or less JavaScript files that are being executed when Nuxt starts, either on the server-side, or in the client side, or both. And you can again do almost anything here. You have access to, say, the router. You have access to generate the Nuxt context. So this is super common to do things like initialize tracking, for example, like initialize Google Analytics or stuff, or also to send a heartbeat, to do like one-off things, or to initialize recurring things. But also adding like a font loader. So there are lots of opportunities there.

And there's also another special part of Nuxt, which is called hooks. And it's nothing – Not to be – Yeah, you shouldn't mix it up with React Hooks. So Nuxt Hooks basically mean you have entry points in the framework processor. So for example, during the build process, there are hooks to say like, "Okay, whenever the build process finished, please like add another file. Or whenever you render a site, please say strip all the JavaScript," because you only want to have HTML for very basic site. Or whenever the generation process finished, please also generate a sitemap based on the size you just generated, which is what the sitemap module, the Nuxt community prides is also doing.

So yeah, lots of opportunities again. And the good part is there are many, many open source modules also like maintained by core members, maintained by the community. I also maintain some. And I update them for Nuxt free now. So yeah, again, you don't have to reinvent the wheel. And that's a huge benefit.

[00:22:02] KP: The idea of generating a site map is particularly interesting to me. How does a plug-in like that get socialized once the developers build it?

[00:22:11] AL: So there are a couple of options. At first, if the developer already built a couple of modules, then there is this Nuxt community organization. And you can pull, I would say, almost any substantial module in there, which means like, again, had the CMs integrations. Things that people need like this item module. We also have the Composition API module in there for like Nuxt 2 to add a Composition API from V3 to V2 and so on so on. So whatever you can do, you can just ask if the module you wrote can be added there. Then of course we take a look. We see, "Okay, it's not like just – I don't know, one liner or something, or malicious." And then you can add it there.

There's also a site called modules.nuxtjs.org, which lists all these modules. So you can also take a look there and search for like a category, search for username and so on. And yeah, eventually there's also a Nuxt newsletter where also modules are being shown. And we also have a Discord where people regularly post cool findings like, again, modules, also websites.

[00:23:11] KP: Are there any common adoption patterns? I don't know, like industries or types of sites that people say, "Oh, yeah. Nuxt is the right framework for me."

[00:23:20] AL: I think I would ask the question differently. I would ask like is there something you cannot do with Nuxt? And I think the good answer to this is, yes, definitely. So if you have a site that's rendered through like PHP or Python, like a traditionally server-side rendered site, then you can't use Nuxt. Because next is the only solution for like the your full like frontend part. You can't just say add Nuxt on top of Laravel or something. You either go like full Nuxt or no Nuxt at all.

For example, for Vue, you can just sprinkle it in, as I also did before in my journey like, I don't know, five or so years ago. But yeah, with Nuxt, that won't work. For everything else, it's fine. You can build classic single page applications. You can build static sites. You can build dynamic, highly dynamic sites. And soon we also have like incremental static regeneration, so ISR. That's also something planned also for Nuxt 3. And yeah, I would say sky is the limit.

[00:24:15] KP: I think you'd mentioned a recent major release. Are there any key features you want to highlight that were recently included?

[00:24:23] AL: Yeah. Right. Nuxt 3 three finally added a beta. So it's not production-ready. That's the first thing I really want to say. We're still working on like a module support, of course, because the core is in beta now. But several modules have to catch up. There are a couple of cool features. At first, Nuxt 3 is finally supporting Vue 3. So the latest major version of Vue. That also means Composition API being supported out of the box. Also with Nuxt composables. So all the Nuxt methods. You had Nuxt 2, but also as composables for the Composition API.

Now, next has like native TypeScript support in terms of it is fully built with TypeScript and ESM. And I think one of the coolest features people also really wanted is there is a Vite integration. So we have extremely fast dev server and using like Vite as a dev server and rollup expander. So that will really step up the developer experience. You can also switch to Webpack 5 if you want to. So we also support that. But yeah, I think these are like many features I want to highlight.

And I think one more tiny thing is that there is a new server engine for Nuxt called Nuxt Nitro, and that one allows us to run on almost any platform. Be it like a service worker or like a Cloudflare networker, or a Netlify edge handler, it's called a thing. It's running on serverless. It's running on just the basic VPS. You name it.

[00:25:45] KP: Wait. That's really interesting. So like Lambda functions, cloud functions deployed there as well?

[00:25:50] AL: Yeah, that works. That works. So I tested Nitro a lot when it was like just in baby steps, I would say. And you basically have a preset for like just having a Node.js server, presets for having a Lambda, presets for having a worker. So not even a Node environment, but just the

V8 isolate as Cloudflare is doing that. And they're like deployments out of the box for, yeah, as I said, Cloudflare, for Vercel, for Netlify. But not only static side, but also as dynamic serverless server-side rendering build. Also Azure functions, Firebase and so on so on.

[00:26:22] KP: So Cloudflare in particular is interesting because they're a globally distributed CDN. All about speed and response time. That coupled with the Nuxt thesis of server-side rendering and all that to begin with seems like there's a real key optimization there. Am I on the right path with fast speed for delivery?

[00:26:41] AL: Absolutely. So, generally, like I would say edge rendering. What's happening in Cloudflare workers is I would say like the next level of serverless. Because now you can just render it super close to customer or to the user. And that means, of course, less latency and more speed. Plus, of course, with Nuxt running there, yeah, you have the best performance you can think of.

[00:27:01] KP: So getting something to deploy in all these places. So the same code can basically run on a Cloudflare worker or on AWS Lambda, but there's subtle differences. They're not the same thing exactly. How do you tie all of these not the same connections together into one seamless interface?

[00:27:19] AL: Yeah. It isn't as easy. Because under the hood, Nitro is doing lots of work there. For example, as I mentioned before, in Cloudflare, you don't have like a full node environment. So you don't have something – Like you don't have access to a file system. You don't have access to lots of node-specific functionalities. So either have to polyfill them more or less if you can. Or if you don't need them, you have to like mock them out. And this is also what's happening. And the good thing is, again, all of this is open source. So Nitro and all underlying packages that do like tiny parts of that are open source. So you can see how all of this works.

And I have to give credits here to Puya and Daniel who mainly work on the Nitro engine. As I said, I tested it and added a few bug fixes here and there. But I'm not the main maintainer here. And they really did a great job. So everything ties together. Everything works great. And again, there are more and more deployment presets coming. And the good part is, as you mentioned, the same code will run wherever, because there is a quite high abstraction level when you write

this code in a Vue single file component, because that Vue single file component, maybe with a bit of Nuxt features used, it will be transformed into JavaScript through the Vue compiler during the build step. And then this JavaScript, of course, can be transformed further to make it work on V8, isolate in Cloudflare worker.

[00:28:39] KP: What's the Composition API?

[00:28:42] AL: So the Composition API came with Vue 3 and was also backported to Vue 2, and it is a new way of writing Vue components, I would say, or like Vue logic. So before we had the Options API where you had like a script tag and then you had just a default export. And you had data. And in there, there was a function that returned like your data. Then you had computed, which was an object that held all your computer properties. So like your reactive parts that React always when, for example, something and your data changed. And you had an option for methods. And you had an option for – Say, you have a function called mounted for mounted lifecycle hook. Yeah, that was the Options API.

And the Composition API had been created because there were a few flaws within Options API. It is super beginner-friendly, but you have one big problem, which is sharing logic. So sometimes you want to share parts of logic between your components, because they're quite similar. And there aren't many options to do this in the Options API. One of them are mixins. But mixins are hard to grasp because it's tough to understand where the actual logic comes from. And it could lead to naming clash and so on.

And also, one other thing of the Options API was you had always a disc call. So, for example, in the method, you want to access one of the data properties in Vue. You had to write, say, disc.user. And disc was especially problematic for TypeScript support. And also for like understanding and getting behind the, yeah, Vue magic, how some people would call that. And that's why the Composition API had been created and now you only had one setup function. And in the setup function, you were able to define all these things you had in your Options before, but without this strict categorization. Now you are able to like group things by feature. So if you have like a feature to fetch a repository, then you can group that together and don't have like a method here and a data there, and a query property here. You all have that in one block. And the good part is, again, if you want to share it now, you can just extract it into a single

JavaScript function, because now there's no like disc anymore. Not that much magic. It's more explicit. And you can share it easily because there's almost just JavaScript. You can move it into a function and then just import it and use it inside the setup function of one two, three components and it will just work.

So this is also, I would say, kind of inspired by, on the one hand, like React Hooks. On the one hand, I think also Svelte was an inspiration there. But yeah, don't judge me if that was wrong. But the good part is that I think it's still quite understandable. It's a bit more explicit. But it's super helpful. And, personally, I use the Composition API whenever I can because I love that style of writing code.

[00:31:27] KP: Can you share some details about your role as the Nuxt.js maintainer? What's demanded of you in that?

[00:31:33] AL: Absolutely. So maybe I can start with, again, a small history part.

[00:31:38] KP: Sure.

[00:31:39] AL: So when I discovered Nuxt because of the SEO demands and so on, I used it, and it was I think close to version two point something. So like half a year before the release there. So I used Nuxt 1. And I was already fascinated, but then I realized some things were not there because, of course, it wasn't that mature in terms of the ecosystem. So I thought, "Okay, maybe let's take a look." Oh, there was an example typo. Okay just send in a pull request. I was kind of familiar with open source due to like Laravel before. So I just did that. And like my changes were very happy, well-received I would say. So I looked into more things and then I found out, "Okay, for one of my projects, I had to add a functionality to add an RSS feed to my Nuxt application." And I took a look and I understood how modules worked. And I wrote my kind of first module, which just added that feed. And then I realized, "Okay, if I needed functionality, maybe there are more people out there that maybe need it too. So I decided to open source the module.

And then of course I was in touch with the people. I moved it to the Nuxt community organization. And then I talked to the core maintainers back then. They were, as I said

Sebastian, and Puya, and Clark. And they were super happy. And then I was, yeah – I read through more and more modules and also the core. And then I started contributing more and more things that I thought might be helpful. And some of them were and some weren't of course. I think that's novel. So yeah, more and more changes were in there, and that was like summer 2018. And then there was the Vue.js London Conference in September. And then Sebastian was like, "Yeah, there's this conference. Do you want to join us?" And I was like, "Yeah, sure." So I booked a flight in the hotel. And they said, "Yeah, the ticket isn't the problem. You can be our guest." And I was like, "Oh, wow! Crazy."

So I went to Ireland before, like a few weeks before to start the internship there for half a year. Then I had to ask for like a vacation, but it wasn't a big deal, and I moved over to London, and I met the other maintainers. So to say, Sebastian and Alex, the first time in real life. And they said, "Yeah, welcome. Thanks for your contributions." So yeah, this is how all the things started.

[00:33:48] KP: Very cool.

[00:33:49] AL: But what's demanded? I thought that it's still interesting. So right now the organization, or like my part is quite loose I would say, because now everything evolved further. I mean, like a few years in the future now. There's a company behind Nuxt now which is called Nuxt Labs and most full-time maintainers. So, again, Puya, Daniel, and also Sebastian, and Alex, they're part of the company. I am not. So I am part of like the community maintainers. That means I don't have that many duties or that I have to do things. But still, I contribute to the core. I try to update my modules. There are meetings every two weeks that I take part of to hear about the latest news and give feedback. And also, of course, I'm in touch with the other maintainers about like problems, experiences. And I contribute whenever I think I can add something meaningful.

[00:34:39] KP: Well, in addition to that, I assume you're pretty busy over at Developmint. Tell me a little bit about what Developmint does.

[00:34:47] AL: Okay. Again, this all started six years ago now when we – So we started with three friends that's just got our A levels, and it was just a funny coincidence that someone asked if we could build a website for them. And I said, "Yeah, we can pay you, but you need a

company." So we founded that company. And we started with just basic projects for small-medium enterprises, so like websites, in-house tools and so on. But especially then when I looked more and more into Nuxt, people asked me if I was able to help them. And I was happy to help them, of course, in my free time, whenever I had time. Say, for example, through the Discord and sometimes also like, "Oh, yeah. There's my application. That's the problem." Like typical like issue triaging, bug reports and stuff.

But sometimes I was like, "Okay, this is super deep and I can't spend a lot of time or free time to look into like projects." And they were like, "Okay. Well, I can pay you." And this happened more and more. And then I thought, "Okay, maybe this could be a good branch for development." So I decided to take the opportunity. And we moved more and more into consulting. And yeah, now that's basically the main part, web development consulting with focus on Vue and Nuxt.js. So basically that means the clients that I have, they asked me, for example, for code reviews for doing peer programming for like discussing architectures and how to build things on a very abstract level. Or also about how to integrate several like libraries, third-party components and so on.

[00:36:14] KP: Do you have any advice for developers interested in taking a similar path?

[00:36:19] AL: Yeah, I do actually. So the first thing is like find a project that you're passionate about that you use personally for a project. And then just take a look and get in touch with the people. And especially if you don't know much about open source, just ask for help. People are super friendly. The open source community is great. And yeah, start with tiny tasks and work your way towards more knowledge and trying out things. Because, basically, everything I did was like I was very passionate. I still am about Nuxt. I took a look into the project and I saw where I could maybe add things, or improve the project, or help out. And yeah, that's why I'm actually here.

[00:36:56] KP: Alex, where can people keep up with you online?

[00:37:00] AL: So I think the best way is Twitter actuall. Sso I have a Twitter handle, it's @thealexlichter, so the Alex Lichter. I also have a blog which isn't that up to date anymore because, yeah, I don't have that much time recently to post. But I promise, I will post a new

thing this year, at lichter.io or at nuxt.xyz. I think that's easier. Yeah, and I think that's it. That are the best ways to reach me.

[00:37:27] KP: Well, Alex, thanks for coming on Software Engineering Daily.

[00:37:30] AL: Thanks for having me. And yeah, I hope you have a great day.

[END]