**EPISODE 732**

[INTRODUCTION]

**[00:00:00] JM:** Wes Bos has created popular courses on React, GraphQL and JavaScript. With hundreds of thousands of students, Wes has earned a cult following for his fun, practical lessons on web development. The course is produced by Wes teach developers how to build useful applications such as a complete e-commerce store while they are learning subjects that are quite useful, such as React or GraphQL.

Wes has built a career around studying and evangelizing JavaScript. His approach to education centers around practice, repetition and hacking on fun projects. He also cohost a podcast called Syntax FM and he is a frequent Twitter user. Wes is a rare mix of developer, teacher, businessman and designer. Throughout his work, there is an artist's sense of attention to detail and a modern entrepreneur's sense of pricing and marketing. His sites, such as JavaScript 30 and React For Beginners, have the deliberate style of someone who has been building websites for a very, very long time.

In today's episode, Wes Bos joins the show to give his perspective on JavaScript, entrepreneurship and podcasting. Syntax FM is a great podcast that I recommend. Lo learn more about Wes's business and his background aside from this episode, I recommend checking out The Indie Hackers Podcast with him, which I put a link to in the show notes. That podcast really opened my mind up to exactly what Wes has done and how much scale there is to his one-person business. It can be pretty inspiring for people. So I really recommend checking out that episode.

Before you started, I want to mention that we are looking for several roles at Software Engineering Daily. You can go to softwareengineering daily.com/jobs to find these roles. We're looking for a few journalists, a podcaster and an entrepreneur in residence. So if you're interested in these, go to softwareengineering daily.com/jobs.

[SPONSOR MESSAGE]

 

**[00:02:11] JM:** We are running an experiment to find out if Software Engineering Dailydaily listeners are above average engineers. At triplebyte.com/sedaily, you can take a quiz to help us gather data. I took the quiz and it covered a wide range of topics; general programming ability, a little security, a little system design. It was a nice short test to measure how my practical engineering skills have changed since I started this podcast. I will admit that though I've gotten better at talking about software engineering, I have definitely gotten worse at actually writing code and doing software engineering myself.

But if you want to check out that quiz yourself, you can help us gather data and take that quiz at triplebyte.com/sedaily. We have been running this experiment for a few weeks and I'm happy to report that Software Engineering Daily listeners are absolutely crushing it so far. Triplebyte has told me that everyone who has taken the test on average is three times more likely to be in their top bracket of quiz course.

If you're looking for a job, Triplebyte is a great place to start your search. It fast tracks you at hundreds of top tech companies. Triplebyte takes engineers seriously and does not waste their time, which is what I try to do with Software Engineering Daily myself, and I recommend checking out triplebyte.com/sedaily. That's T-R-I-P-L-E-B-Y-T-E.com/sedaily. Triplebyte, byte as in 8 bits.

Thanks to Triplebyte for being a sponsor of Software Engineering Daily. We appreciate it.

[INTERVIEW]

**[00:04:09] JM:** Wes Bos, you are the creator of several popular online courses about JavaScript. You're the host of Syntax FM, a popular podcast about frontend development. Welcome to Software Engineering Daily.

**[00:04:20] WB:** Thanks so much for having me. I'm excited to be here.

**[00:04:23] JM:** So you teach these popular courses on frontend technologies, and your courses center on building apps that feel like real world use cases. For example, you have a detailed e-commerce application that you build from end-to-end. This is in contrast to some other

approaches of software engineering or computer science education that focus on the theoretical approaches, but your focus is more on applied to software engineering rather than computer science theory. I guess it's not exactly just frontend, because it's full-stack. It's a frontend and then a node backend, for example. What led you to focus on this format of applied software engineering rather than the computer science theory?

**[00:05:08] WB:** Yeah. I don't have any sort of computer science background. I went to school for what was called business technology management. So it's a business degree focused on tech and like management servers, and I majored in telecoms. So I can tell you how to calculate IP address if you need that. But other than that, I'm completely self-taught coding, because they didn't actually take any coding classes in school. I always found it extremely frustrating when you'd see these examples of foobar baz and all these stuff, and I just be sitting there being like, "When would I ever have to use this myself?" I've also taught a lot of people in person as well. I'm not saying that this is everybody, because certainly people come out of the woodwork anytime I say this. But I know that at least the type of person that takes my course much rather see this as how it's done in person and this is where you would actually use something, and I'll sneakily teach you the concepts behind everything along the way.

I'll teach you why we need to use MapReduce filter. I'll teach you why we might want to improve the flow of a function or not have mutations or things like that, but in the context of an actual application that you would build. So that's sort of been my whole mantra. I don't use foobar baz. I don't like to do too much of just sitting there and writing loops or count logging things. I certainly do that when you need to teach a concept, but I like to apply it as much as possible.

**[00:06:36] JM:** You cover this full-stack JavaScript education. The idea of a full-stack developer is much more popular today than it was in the past. So when I graduated college five or six years ago, me and most of my classmates categorize ourselves explicitly as a backend developer or a web developer and we kind of die-cast ourselves as that role so that we would slot in to some team at some corporation, but it seems like there's something this change this led to the growth of full-stack development or comfort with the signing yourself the role of a full-stack developer. Was there something that changed in web development that lead to the growth of full-stack development?

**[00:07:25] WB:** That's a good question. I don't really know, because like when I first got into this stuff, it was just really out of necessity that I wanted to make things, and in order for me to make my thing, I needed to learn both sides of it. I needed to learn how to build at frontend. I need to learn also how to build on the backend.

So that was purely out of necessity on my end, but I think like why is it becoming much more popular? I think first because, at least in my circle, node is getting extremely popular and the fact that someone who is pretty good at writing JavaScript on the client-side can also pick up and continue to write, because that was a huge hurdle for a lot of people. I just don't know the language that our backend is written in, and there are obviously other hurdles, like understanding how do server works and what are requests and what are streams and all that. All that stuff that goes along with understanding how backend works.

But if you take the hurdle of the language as something I already know, so there something right there. I think also what's starting to happen now with a lot of applications is a lot of the logic is actually happening in the client. A lot of the heavy lifting happens in the application itself, whether you're building something, like React, or whatever, and your backend is starting to become more headless, where it's just fetching data, massaging that data before it sends it, handling all the requests, handling all your authentication, things like that.

I'm not going to say that backend is getting easier, but I think that it is definitely much more approachable for someone to get in there and start to build and sort of make things work themselves if they don't necessarily have a whole lot of backend experience.

**[00:09:03] JM:** Who are the typical types of students that are purchasing your courses? Are they brand-new to software engineering? Are they looking for some continuing education? How is the prototypical student look?

**[00:09:15] WB:** It's actually all over the place, which I'm surprised. I do have a lot of people who are new to programming in the last year or two, and that certainly probably the people who I interact with most, because they're the people that ask a lot of questions and are really involved in the community. But also a huge portion of my audience is just existing developers and existing teams. Like I've got – Like you name any large company, I bet that I've sold them some

sort either a team license or a couple of licenses to some of their employees. Just because the JavaScript moves so quickly, that there's often good developers that just need to figure out how this stuff works and they just grab my course, they run through it, they're good developers already. They understand how these all works and they're able to build the thing that we build in the course and then apply that to whatever it is that they're building at their company.

It' something that surprise me, because like my audience is not necessarily beginner. I don't have any beginner-friendly stuff just yet. So I would say it's probably more intermediate. But then again you do see people all over the map that are looking to learn this tech.

**[00:10:20] JM:** That's interesting what you say about the corporate training side of things. That's such a big business. I know Pluralsight makes a lot of money off of the developer retraining, enterprise sales. Same thing with O'Reilly. I think O'Reilly's biggest business now is the Safari Books online and they get these renewing enterprise licenses that every large tech company has purchased on a renewing basis. What's been the process of expanding into that enterprise sales model? Because it kind of seems like maybe you started as just kind of putting up these courses and individual developers, but maybe you expanded to the company model over time?

**[00:11:03] WB:** Yeah, it was just a matter of developers inside of these companies being like, "Hey, do you have a team license?" So I didn't at first. So I just coded that into my platform and people started buying them. There's real like enterprise sales process that I have. All it is just there's developers that are inside of these companies and they've got so much to spend every single year either on conferences or training, and they like my stuff, because they listen to the podcast. They've taken some of my free courses, and they need to learn a specific tech. So they go ahead and buy it.

It's kind of frustrating to –I know that it's frustrating to companies, because they often have these subscriptions that they're paying dearly for to these things like Safari and Pluralsight and then their developers come say, "But I want Wes's instead," which is always pretty funny. So I often have to write little emails to their managers being like, "These are little topics covered in it," and they likely – I don't know. I'm not saying if they're not covered, but usually the employer

is saying like, “I need to learn this and it's not specifically covered in the thing we’re already subscribed to.”

**[00:12:12] JM:** Well, since you are deeply familiar with the changing pace of frontend, I do want to ask you some about some different aspects of frontend development. Frontend engineering changed really significantly with React. There was finally consolidation around a frontend framework after there had been some fragmentation between Backbone, and Angular, and I think there were some other frameworks around that time. What do React get right? What do you think caused the consolidation to finally happen around a frontend framework?

**[00:12:45] WB:** Oh! That's a good question. I think that it was a bit of luck. I thing Angular kind of handed it to them, because Angular 1 was super popular at the time. Backbone was still pretty popular. Some people were using Knockout. Ember had just come out. But Angular announced like way, way, way ahead of time, I think it was like a year or a year and a half that, “Hey, we are totally redoing Angular, and Angular 2 I not going to look anything like Angular 1.” That was me at the point. I was like, “Oh! Well, so should I stop using Angular,” or like at the time it was kind of poorly announced where they said there'll be no upgrade path or something like that and they've, of course, figured out an upgrade path. But there was like thousands of developers, myself included, being like, “Well, I don't really want to continue investing my skills in this thing that's going to be dead in a year, or a year and a half, or whenever it comes out. So maybe I'll just try some of the alternatives.”

So I tried our React and I really liked the way that it worked. I really liked how simple it was. I think that it was relatively simple to learn at the time because it didn't – And it still doesn't. It doesn't include anything, like you don’t have to relearn how to fetch data. You don't have to relearn how to do like routing or you don't have to relearn all of these. Like Angular always had like the Angular way to do something, and React was simply just like the way to handle your templates. If you want to fetch data or manage your data or do routing or anything like that, that’s not part of what React does, and you have to reach outside and likely pick up some of these libraries that you're already familiar with.

**[00:14:20] JM:** Have you worked with View?

**[00:14:21] WB:** I haven't. It's kind of a funny ongoing joke on our podcast, because that's like probably the most common question that I get, is what are your thoughts on View? I've written a couple little things in it, but not enough to talk about it too much. But I do know that it is – I would say that it's even simpler then React. It kind of takes the things that we loved about Angular 1 and marries them with the things that we love about React. Then it also gets rid of some these like weird things that we have in React that are sort of unnecessary to know.

So I'm a big fan of it. I think that it's going to continue to become extremely popular. Also, View is huge in China and a couple of other countries as well. I know we don't hear about it maybe as often as we do, but some other countries, it's like the framework.

[SPONSOR MESSAGE]

**[00:15:14] JM:** Managed cloud services save developers time and effort. Why would you build your own logging platform, or CMS, or authentication service yourself when a managed tool or API can solve the problem for you? But how do you find the right services to integrate? How do you learn to stich them together? How do you manage credentials within your teams or your products?

Manifold makes your life easier by providing a single workflow to organize your services, connect your integrations and share them with your team. You can discover the best services for your projects in the manifold marketplace or bring your own and manage them all in one dashboard. With services covering authentication, messaging, monitoring, CMS and more, Manifold will keep you on the cutting-edge so you can focus on building your project rather than focusing on problems that have already been solved. I'm a fan of Manifold because it pushes the developer to a higher level of abstraction, which I think can be really productive for allowing you to build and leverage your creativity faster.

Once you have the services that you need, you can delivery your configuration to any environment, you can deploy on any cloud, and Manifold is completely free to use. If you head over to manifold.co/sedaily, you will get a coupon code for $10, which you can use to try out any service on the Manifold marketplace.

Thanks to Manifold for being a sponsor of Software Engineering Daily, and check out manifold.co/sedaily. Get your $10 credit, shop around, look for cool services that you can use in your next product, or project. There is a lot of stuff there, and $10 can take you a long way to trying a lot of different services. Go to manifold.co/sedaily and shop around for tools to be creative.

Thanks again to Manifold.

[INTERVIEW CONTINUED]

**[00:17:28] JM:** I feel like there might've been some kind of – There's some kind of cultural affiliation with these different frameworks. There's something about Angular coming out of – It came out at Google world, right?

**[00:17:40] WB:** Yeah.

**[00:17:41] JM:** There's something about the Angular coming out of Google world, and then React coming out of the Facebook world of this ruthless pragmatism and kind of a feeling of just pragmatism that sometimes looks like sloppiness, but it's actually this pragmatism. Like JSX just looking appalling to people at first and then overtime realizing it was really, really useful. Then View coming from this world of this one guy setting up this framework, but having kind of a vision for the design of the framework and the experience of the developer who is using it. Then it seems like to some degree it's kind of interchangeable between them, but except for the fact that you have these network effects and then you get all these education and stack overflow post and stuff around the biggest game in town, which is React. Then it kind of becomes a self-fulfilling prophecy that it becomes the "best framework", just because it has the most resources around it.

**[00:18:43] WB:** Yeah. It's kind of amazing that View has such a high-market share given that it was initially just crated by this one guy and now is used by – I think Alibaba is a heavy user in View and I know a couple other large companies use it as well. So it is kind of interesting seeing where people have come from.

There was a time where the future of React looked very grim and View is gaining tons of popularity because of the – I'm not sure if you member the patents clause that React had, where Facebook could like own all your whole company and your children and everything if you wrote one line of React, and people were freaking out about that. Thankfully, they came out and relicensed that so that it wasn't otherwise, because I was getting emails like all day long, "What are your thoughts on this patent thing?" It was very frustrating, because like Facebook is not going to come and take your family. But the way that their license was worded looked like they would.

**[00:19:40] JM:** Yeah. Well, I think they have done things in the past that I think have disrupted some level of trust with developers or with their community. I guess – Well, whatever. It's hard to earn trust.

**[00:19:53] WB:** Totally. Oh, I agree. It is funny that React is from Facebook, because it's probably one of the least trusted companies in most people's minds right now, which is like nobody likes Facebook right now. But I'm thankful for the nice little framework we get out of it.

**[00:20:08] JM:** Yeah, and I really like the team. The team seem really open and interactive.

**[00:20:12] WB:** Yeah, they're amazing.

**[00:20:13] JM:** As the React ecosystem is matured, how has React development changed, or has React development changed? Has it just been plumbing under the surface that's getting more efficient, or has the actual writing of React apps evolved?

**[00:20:29] WB:** Yeah. There was some big updates about a year or a year and a half ago, React Fiber, and that was just a total under the hood rewrite of it and it just got much faster, and the API looked exactly same. But in the next coming months, we are seeing a pretty big update to what the API looks like. Not rewrite, because everyone sort of groans when things change. They go, "Oh! Of course, I have to rewrite my entire application." But we're just getting more APIs that will allow us to write these things in kind of a nicer way. The two big ones are – The first one is called Suspense, and that's going to allow us to write these interfaces that rely on a synchronous data, because that's the big thing in the browser, where the thing loads and you

throw up a spinner and then you go off and fetch some data and you come back, and before you know it, it got like 15 different spinners in your application and the whole thing is just flashing back and forth as the data loads into it and you have to rewrite it to that component.

Suspense is going to make that much easier. It's kind of a really neat API where it almost looks synchronous, where you can just fetch the data and then render it out and there are some rules where the higher components will be able to detect if lower components are in this suspended state. Then the other one is this new Hooks interface. So React adopted classes, ES6 classes, probably about two years ago, two, maybe a little bit longer than ago.

One of the downsides to React classes is that we don't have properties inside of the classes, like instance properties, which was a huge pain if you ever want to put a method that is balanced to that instance. So there's all these like workarounds where you can use this like ES6 thing, or you can use this like proposal and an arrow function that will bind to the instance.

They've come out with this new thing, which is called Hooks. Hooks are going to allow us to just not – You can still use classes, but you can just write a function that returns some JSX, and if you need to hook into things like state or context or the event cycle event, event methods which are like component will mount and component did update and component will un-mount. If you need to hook into any of that and do some side effects, you can do that fairly simply just with these nice little functions that are called Hooks.

**[00:22:51] JM:** Your frontend teaching – I should stop saying frontend. I was going to say web development, your web development teaching.

**[00:22:58] WB:** Yeah, I don't care what you say. Sometimes people get all out of shape about frontend versus backend. But yeah, we're just making websites and web apps here.

**[00:23:06] JM:** Yeah. I think it's because it's like you're not spinning up Kafka clusters and Kubernetes and VM's and stuff. So I don't know. Anyway, so your teaching also includes GraphQL, and GraphQL is a system for unifying data access and making it simpler and faster for developers to issue these complex queries that hit a lot of different data sources. How has GraphQL changed how you think about web development?

**[00:23:34] WB:** Oh! That's a question. I think that it makes – GraphQL makes attacking certain queries that could be complex, very, very simple. So I don't think that it has really like changed the way I think about web development. But I just know that it's going to be a lot easier when you build a complex application with GraphQL. It's no sweat to make a whole bunch of different types of models and link them together. Whereas like maybe before where if I had a bunch of different pieces of data, I wouldn't want to like have to deal with crating multiple models and all the relationships between them and two-way updating and all of that sort of headache that comes along with that.

So GraphQL and a lot of the tooling that supports GraphQL, because GraphQL is just a spec and there's a whole lot of tooling that needs to happen on your server underneath the hood to actually do those relationships. But it's really cool, because it's going to allow us and it already does allow us to do a lot of these queries right from our client in the browser, or you can run GraphQL wherever you really want without having to worry about setting up endpoints and getting all the data and then maybe waiting for three pieces of data and joining them together and making sure they all relate to each other, which has always been a bit of a pain at least for me in the past.

**[00:24:55] JM:** I know that GraphQL is useful for these gigantic apps with lots of data sources, so obviously coming out of Facebook. Is it also useful for smaller applications? Should people just start with GraphQL from day one with their small applications?

**[00:25:10] WB:** Yeah. I think that it does work really well for small applications, but there is an overhead which is setting everything up. There is an overhead which is setting up your – You have to set up your schema still. So you can either have your existing schema in your database or you could have like an existing data source that's coming from somewhere else and then you have to set up your resolvers on your server and then you also have to set up the queries on your client-side. Whereas in the past, it used to be like – I don't know, spin up an endpoint, hit that endpoint, dump some data and we'll catch that in the client and run with it.

So there's definitely a lot more of a set up with GraphQL and there're a lot of tools out there that this might not be the case in six months, a year. There are a lot of tools and people out there

trying to make this a lot easier than it is right now, which is pretty exciting. So should you use it on your little project, I probably would say not unless you know that you're going to have a whole bunch of relational data. I think you'd probably reach a point in the project where you go, "Oh! This is probably worth me spinning up a GraphQL server and switching over to that right now," versus, "Oh! I've got a weekend. I want to build some fun. I just want to hack it together." Should you go through the setup of it? I probably would say no. Not just yet.

**[00:26:32] JM:** Yes. You mentioned this tooling. There are some different GraphQL open source tools. There's Relay, which is just Facebook's original open source GraphQL. What is it? It's a GraphQL server? What does it do again?

**[00:26:47] WB:** No. There're a couple pieces that you need if you're doing GraphQL. You need something on your server that will serve up that will respond to these GraphQL queries with the appropriate data, and how these data get out of your database or from the data source, those are called resolvers. So you first need something on your server that will run that. Whether you just write your own or there's a lot of like kind of – There're two big ones. There's one called Hasura and then there's one called Prisma. Prisma is the one I use in my own course, where they do the whole database API relationship schema, all that stuff for you. Then you need something on the client-side, which is actually going to be making these queries. The two big ones out there for client-side development are Relay, which comes from Facebook, and Apollo, which comes from folks who are behind Meteor, for all those years.

**[00:27:40] JM:** Okay. Right. So if I'm a frontend developer, I'm going to make a GraphQL request to the GraphQL server that's going to interpret that query and hit all the necessary data sources to return my one unified request. So in that entire query process, you need something on the client that's helping you issue the query, which is Apollo, or Relay. Then on the server side, you have Prisma, or Hasura you said?

**[00:28:13] WB:** Hasura, yeah. I think it's built in – Just double check. It's built on PostgreS. Prisma will take in your schema and go out to I think it's MySQL, or Mongo, or PostgreS. It will work with all the different databases. Hasura just sits directly on top of PostgreS. But then you still – At that point, you still need a logic layer at that point, which is going to handle your file

uploads, all of your authentication. Any extra logic that you want on there pass your CRUS operations of these data pieces.

**[00:28:50] JM:** Right. Okay. So this is useful, because – Well, the reason you want these resolvers is because the GraphQL request doesn't look like a PostgreS query, and so you need some layer to interpret that GraphQL request, which looks kind of like a JSON object into like a PostgreS query or whatever, MongoDB query, or something.

**[00:29:15] WB:** Yeah. That was one thing that really tripped me up at first, is GraphQL is not really a query language, because it doesn't actually do anything for you. It doesn't query your database. It doesn't do any filtering. All it does is allows you to make requests, which are either queries or a mutation, and then it's up to you to figure out how do I complete this query or mutation under the hood? To a lot of people that's really frustrating, because you have to define your schema in GraphQL on your server. You have to define your schema in your database. You also have to define your schema in the client-side, and those things are often the same, but sometimes are different. If you want to request a user, you don't want to be able to surface their password to the client-side, but you do want that data on the server-side, right?

**[00:30:02] JM:** Right. So we did a show with Prisma not too long ago, and I was trying to understand how big is the market for tooling around GraphQL, and is there – What would the business – I mean, they kind of have an interesting situation where obviously, okay, there is room for tooling around GraphQL. You're using you use Prisma yourself. But then there's a question of like, "How can you build a business around that?" They're faced with the interesting question of how do you take that query interpretation layer and turn that into a business. What do you think of the business around GraphQL tooling? Is there a market there?

**[00:30:43] WB:** Absolutely. I think this comes back to the first question you're asking me, is like why are people becoming full-stack? And it's because of services like this. The React developer who is and knows how to build a wicked frontend application doesn't necessarily want to be bothered with setting up the entire – Because most of these applications are just like CRUD operations. We're just saving data to a database, pulling it out and all of that.

So I think that having something do it for you, I'm not sure whether it's something like Prisma, or Hasura or something that you throw up on a Docker image yourself. I think that is going to become really popular. We already see it with services like Firebase and Parse and whatnot. If you can just like take that whole part away for me, I think that's great. I think even further, we'll probably see in the next little bit is we'll see entire things where people put the user accounts and authentication on top of that as well, because that's one thing you currently still have to roll yourself.

**[00:31:46] JM:** Amazon came out with something recently. This some kind like backend as a service suite of tools that came out of the – Because they acquired a GraphQL company. Have you followed the AWS tooling world in terms of have GraphQL?

**[00:32:00] WB:** Yeah. The name of that is – I was just looking at it today. Hold on. One sec.

**[00:32:03] JM:** Amplify. Is it Amplify?

**[00:32:05] WB:** Yeah. AWS Amplify. That's what it is. I think that's pretty interesting. I haven't really looked at it all that much. Just there's so many things that are in this space right now. But it definitely is interesting to me. If you look at Firebase, they're trying to do this as well with like serverless functions and they also do your authentication. The also do your real-time database. It's kind interesting. I definitely see that people are going to be able to build apps on top of these platforms, because they don't necessarily want to or need access to the whole layer underneath it.

**[00:32:39] JM:** Yeah. I've used Firebase a number of times, and I really, really liked it. So you got Firebase and also Netlify in terms of the really simplified backend hosting kind – I mean, I guess – I don't know. Do you consider Firebase and Netlify in the same category?

**[00:32:57] WB:** Well, Netlify is more of like you push your – So there's this whole idea of called like the JAMStack, where you just have a — It stands for JavaScript APIs and markup. You write your React application. That will talk to a server somewhere and come back. The idea of Netlify is that you can host your entire application on Netlify, because it's just HTML, and JavaScript,

and CSS. They are also getting into to serverless functions as well because that's going to be helping – That's going to help do the whole API endpoint of it.

I don't necessarily know. I haven't looked too much into the Firebase hosting side of things. So they probably are competitors at some point. But Netlify, at least in my books, is like the answer to like where do I easily host up a React project or one of these like single page applications without any of the headache that often comes along with trying to configure like nginx, or Apache or something like that on your existing bugs.

**[00:34:00] JM:** Right. There's something about the Netlify path to deployment that's really simplified and it's just – Okay, we just can't do this one thing, and then there's other things that we do off to the side of that, but you come to us for being able to have a git workflow and pushing to hosting and having this really simplified path. Whereas Firebase, I think they do have a hosting, but maybe like that one simplified thing that you go to Firebase for is more – You want to real-time database, and they do have hosting, but I think the on-boarding path to that is less clear than in terms of the hosting than probably Netlify's direction.

Yeah, I was at the JAMStack Conf, and we're going to be up there and I think our paths didn't cross, or I don't know. I wasn't there. I was only there for one day. But it was an interesting conference, because this was a side of frontend development, or whatever – Full-stack development. That I really have not seen much firsthand, because a lot of the shows that I do, I'm talking to like an engineer from Netflix, and it's like Netflix it is probably not using the JAMStack because they're like building Netflix. Have you started to see people building full-fledged businesses or like startups built on the JAMStack, or do you think it's still very nascent?

**[00:35:22] WB:** That's a good question. I don't know if I just like look at anybody's website. I don't know if that is JAMStack or not, because I don't open up the terminal. I can't give you any examples of what it is, but what I think, I'm going to tell you one thing. I was looking yesterday at the Gatsby documentations. So Gatsby's kind of like a React-based, I don't want to say static site generator, because that has like the wrong connotation, but like it's the ability to make a website with React.

Their tutorial for learning it is very much aimed at setting up your development environment and you should use Prettier and you should use these extensions on VSCode, and here are some other things. Like these are not anything related with Gatsby, but they're related with becoming a web developer.

So what I thought was cool about that is that they are obviously targeting what the WordPress developer was eight years ago, where the person's first entry into web development at the time was WordPress, and then they started hacking away and where it'd be able to build a website. I think that's what a lot of these companies are targeting right now just because these are new people learning a new way of building applications, and maybe in a couple of years as they become intermediate advanced developers, this is going to be the way they know how to build stuff. It's much like you talk to people who've been doing WordPress for 10 years. That's the way that they know how to build stuff and that's how they always solved their problems.

**[00:36:52] JM:** You're pretty familiar with the WordPress world, right? What did you think of that rollout of the new – Because they updated their kind of interface and they had this strong push towards React, and there were some turmoil in the community around that. But you got to be sympathetic to Matt Mullenweg to some degree, because the platform is – It's an older platform, and he's got to do something to push it forward. What did you think of the decision to – Or maybe you could just retell what exactly happened in the WordPress community and your thoughts on it.

**[00:37:30] WB:** WordPress has this. They've been using this thing called TinyMCE for their editor and they've given you one box to input your data, and the inside of that box you can use an H2 and a paragraph tag and you insert image and all that stuff. If you wanted a better experience, like people are flocking to Medium for a while there, because Medium has a really nice writing experience. So they made this thing called Gutenberg, which is – This idea is that you're allowed to great these Guten blocks and you can create these different sections of content and move them around and create your own custom input Guten blocks and all that and it's just kind of this new authoring experience to WordPress in general.

That whole authoring experience is rich. So it needs to be built in something like React. Thankful, they wrote the entire thing in React and then the patent stuff came out. Facebook

wasn't budging on the patent stuff, and then that Matt Mullenweg way came out and started to flex and he said, “All right. Well, better rewrite this thing in View,” and then a couple of days later Facebook came out and fix the patent stuff, which was good. So that was a bit of a side.

I think it's really exciting that we are getting a better authoring experience. I know from what I’ve heard from users is that they are much happier with it. So not necessarily the developers, but the people who are using it on the other end, they’re much happier. The people who I see who are unhappy with it, first, there's apparently some accessibility issues with it, which is silly that they haven't been addressed yet, because that’s not something you should ever do to anyone using your website is make it inaccessible. So that’s one thing. I think that can probably be solved though. I think the bigger thing is that in order to work with this thing, you need React. You need to know React. I know that in the WordPress community, there’s a lot of really good developers. But I know that there's a lot of more like hobbyists, where they know how to make a pretty good theme and they have theirs their plug-ins that they’ve built. There's a huge ecosystem all built on PHP and all these plug-ins built on this old editor. Then this is now getting shaken up and it's leaving some people out, because they’ve probably tried to learn JavaScript many times and it's a frustrating language to learn. It's pretty hard.

I think that's probably where a lot of these pain points are coming from, where people are frustrated that they have to relearn something. WordPress has been very stable for the last – I don’t know, 6, 8 years. I have my own WordPress website I just updated. The other day, I just had a Yolo updated like 20 of my plug-ins and nothing broke. That's unheard of in JavaScript land. Everything will break every week in JavaScript land.

I think that that's probably what it is at right now. There’re a lot of frustrated developers that they have to relearn. It's a different experience. I'm sure there're lots of other little valid concerns with it as well, but I think on the whole, it's good that they're doing this.

**[00:40:25] JM:** What is it about WordPress that has allowed it to stay so dominant for so long?

**[00:40:32] WB:** I think it’s easy to learn. It's was my first entry into web development. It’s a lot of people's entry. You can get something up and running really quickly, huge community of

WordPress developers, of plug-ins, of meet-ups. If you ever go to a WordPress conference or anything like that, these are some of the nicest people you'll ever meet in your entire life.

I think also just like ease of deployment. You can just drag-and-drop the sucker up on to a server, hook it up with a MySQL database and you're up and running. So I think for the longest time it was the best solution, and I think it still is one of the best solution, because like I think headless WordPress is going to be something that is huge in the next couple of years, where you can still build your website in React if you want and you just use the WordPress as a server that you can pull your data from.

**[00:41:25] JM:** Have you spent much time working with Gatsby? Do you have any perspective on how you think Gatsby is going – So Gatsby, as you said, is this – I guess a headless way of deploying your websites, or how would you describe it?

**[00:41:38] WB:** So the best way to explain it is it's like a static site generator. So you get React.

**[00:41:42] JM:** Static site generator, right?

**[00:41:43] WB:** You get your router. You get all the way to the link between pages. Also, right off the bat, it's going to be like prebuilt. So it's just going to be really, really fast. Their service worker is built-in. There's a lot of plug-ins for things like compressing and doing smart things with images. There's a lot of plug-ins for bringing your own data. So if you want to attach that to a WordPress, or markdown, or any GraphQL API, you totally can. That's the whole idea. So I definitely think that Gatsby is – I'm going to call it like the next WordPress. The whole like authoring end of it definitely still needs to be solved. You can obviously bring your own. You can bring your own – You can still use WordPress, but I think it's just going to be making a website really easy and making a really good and fast website really easy.

[SPONSOR MESSAGE]

**[00:42:41] JM:** OpenShift is a Kubernetes platform from Red Hat. OpenShift takes the Kubernetes container orchestration system and adds features that let you build software more quickly. OpenShift includes service discovery, CI/CD built-in monitoring and health

management, and scalability. With OpenShift, you can avoid being locked into any of the particular large cloud providers. You can move your workloads easily between public and private cloud infrastructure as well as your own on-prim hardware.

OpenShift from Red Hat gives you Kubernetes without the complication. Security, log management, container networking, configuration management, you can focus on your application instead of complex Kubernetes issues.

OpenShift is open source technology built to enable everyone to launch their big ideas. Whether you're an engineer at a large enterprise, or a developer getting your startup off the ground, you can check out OpenShift from Red Hat by going to softwareengineeringdaily.com/redhat. That's softwareengineeringdaily.com/redhat.

I remember the earliest shows I did about Kubernetes and trying to understand its potential and what it was for, and I remember people saying that this is a platform for building platforms. So Kubernetes was not meant to be used from raw Kubernetes to have a platform as a service. It was meant as a lower level infrastructure piece to build platforms as a service on top of, which is why OpenShift came into manifestation.

So you could check it out by going to softwareengineeringdaily.com/redhat and find out about OpenShift.

[INTERVIEW CONTINUED]

**[00:44:50] JM:** You teach JavaScript, but you have this broader set of skills around design, and business, and marketing. I was a spending a lot of time browsing your websites and the different courses you have and the design of them, the attention to detail is really well done and it's just beautiful. It speaks to –There's certainly an artistic sense there. But also a well-honed sense of who your audience is and kind of the marketing and the business elements of it and the pricing. I think the pricing seems really well calibrated. What advice do you have for developers who want to complement their engineering? Because you have a very deep engineering skillset, and I think on the Indie Hackers podcast, which was really good by the way, with Courtland, he asked you like, “Of all these different skills that you have, which one would could you not do

without?" You said software development, of course. But you really do have – You have the depth of the software develop, but you have like kind of the T-shaped, you have the competency and the design and business and marketing and so on. What advice do you have for developers who want to complement their engineering skillset?

**[00:46:06] WB:** Oh! That's a good question. I think that the marketing stuff for sure can be learned. It's just the whole marketing space is so icky. I think that the best way to describe it, because you go out there and you start looking for marketing advice, and it's all stuff that you hate being done to you, because we're developers and we know how this stuff works. It's is very hard to trick us into things like that.

So I think that the marketing stuff can be learned just by knowing who your audience is and knowing what works well. I think the reason why it works so well is because I just ask myself, "How would I market this stuff to myself?" Then you also listen to things like Indy Hackers podcast and there're lots of like a really good tips from people's businesses that are outside of what I do. So I'll give you an example. My most popular core, which is javascript30.com. It's a free course, and I have I think over 200,000 people have signed up for this course. How I got that, I was listening to a business podcast about a green smoothie challenge where these two girls have this thing called the green smoothie 30-day challenge and they got their green smoothie girls and they send you like one a day and you go buy the ingredients and you make a smoothie and that's the challenge. I'm like, "That's awesome!"

**[00:47:19] JM:** That sounds fun.

**[00:47:19] WB:** I bet that would work really well for JavaScript, and it did. It did work really well. There's obviously a lot more to why that course did so well. But think that there's something to be said for don't like look at what are other, because I see people all the time just copying exactly what I do like down to the FAQs, people have copy-pasted it. That's not going to work. I've already done that. I know my audience and why I've done that, but if you look elsewhere, and what is working well in other industries and figure out how do I apply that to what I'm trying to do here. So that was marketing.

Design, I started off as a designer many years ago and I've always just had I guess an eye for that. Like I definitely wasn't good when I started, but I've always enjoyed trying to design things. Also, when you're building a course, it's tough to build a course for eight months and being able to stop writing GraphQL queries and pick a font and a color scheme and the right texture is very, very nice when you are so sick of writing JavaScript all day long.

**[00:48:22] JM:** Did you do that green smoothie challenge or did you hear about it?

**[00:48:27] WB:** I didn't do it, no. I just heard about it.

**[00:48:29] JM:** Okay. I have made some green smoothies and sometimes they taste really good. Sometimes they taste awful, but usually it's like you have to put – It starts to get really sugary when you put more fruit in it. Anyway, that's a different topic.

**[00:48:42] WB:** Yeah. We've done green smoothies, and my favor is lots of ginger. That's not full of sugar, but it makes it taste great.

**[00:48:49] JM:** That's right. That's right, ginger, or like lime. You could some lime or lemon. So the job marquee is changing and developers can work remotely. They can do consulting through Fiverr or Upwork or they could stand up their own company, and is much easier to start a company today than ever before. There seem to be more solo developers than ever before. That's the whole thrust of Courtland, Indy Hacker thing, and then as you have done, you can learn these variety of skills that you need to thrive as an individual.

Have you experienced any cost of that, like that the experience of being a solo developer? Do you feel like you're missing out by not working on a larger team? I mean, obviously you have people you work with and various partners, but it's not really like a software development team as far as I can tell. Do you feel like you're missing out at all?

**[00:49:47] WB:** I don't feel like – Like I know I am, for sure. There're things like scaling up a server, or like I was asking the other day on Twitter. I was asking for recommendations on kind of like self-hosted Heroku and there're all these things where when websites get to a certain point, I probably would love it. But like I would love to work at Netflix for a little bit, because I'm

sure there's a lot I could learn about building really big websites or problems that they've run into, because when you work on a team and you work at specific types of businesses, there's things that you will run into that you will never run into as an independent developer. I've worked with many companies especially as a consultant. I've seen a lot, but I'm sure there's stuff that I'm missing out on it. But that doesn't really bother me just because like I do come across a lot of different situations that I need to solve in my own course platform and when I was doing consulting. I know that I'm not trying to know absolutely everything. I know that I'm just trying to be pretty good at React, in Node.js and that's my lane that I try to stay in.

**[00:50:52] JM:** Yeah, and I guess you solve the isolation part of your business – Well, you have a family, but also you go and do teaching in-person. Because I think isolation, that's actually one of the reasons that drives people to not want to do remote work as much.

**[00:51:11] WB:** Yeah. I actually don't teach in-person anymore other than just like workshops once every couple of months. But I love being alone. I'm on Twitter all day long. I'm on Slack all day long. If I did wish anything else, it would be that more people would leave me alone. It's not that I'm very extroverted and I'm the life of the party when I'm out with my friends, but there's something so sweet about not having meetings and not having questions from everybody and just being able to open my calendar, and there's nothing on my calendar for the day and being able to just hack away and work on whatever it is that I want to get done.

**[00:51:49] JM:** I'm totally with you there. The first year or so starting this podcast, there was like – I was like, "This is just great." I'm just grinding and I can kind of make my own schedule and so on. But I reached this point where I was like, "I think I'm going slightly insane," because I'm just spending all my time in my apartment with my cats and like barely go – Because you're just trying – Part of you, or at least for me I was just like, "I want to own my own thing and I want to be able to have my own thing and I'm just going to it focus and I can get completely absorbed in it," but you lose – If you don't go out enough or if you just only go to the coffee shop and whatever, personally speaking, I began to like lose a sense of proportion in the world, because I just wasn't interacting with enough people and I really had to strive to do more interaction with other people. Maybe you have friends or something.

**[00:52:38] WB:** Yeah. I've got lot of friends. We go out for coffee every now and then, see them on the weekends, talk to people. I also do my own podcast. So I talk to Scott, my cohost, for three or four hours on audio all week long. So people ask me about that all the time, and I can say that that's never been an issue for me.

**[00:52:58] JM:** Okay. Let's talk about the podcast more, because you host Syntax FM, which is a popular podcast about web development. What makes for a good web development podcast?

**[00:53:10] WB:** I think that we started it off, because – No offense to your podcast, but we have found many web development podcast, like that we're very good. There's lots of them out there, but they're either dry, the audio is terrible, they're not out frequent enough, they're not all that interesting, at least to us. We kind of said, "Let's do a podcast," and a couple of our key points were it needs to be helpful. So it it's not just us sitting around shooting the shit about how things work. We actually want to make it almost like a tutorial, the podcasts.

That's why we call it a Tasty Treat Podcast, because we want you to leave listening to the podcast going, "Huh! I actually learned a couple of things listening to that podcast. That was really helpful," or "I had no idea what course was, and I just listened to a 25-minute podcast and Wes explained what course was." Just like stuff like that, or I get emails like all the time people just have questions about things. It's so nice to just be able to point them to a podcast and say, "Listen to this. It will fill you in on everything that's going on." So that was the first one.

Second one was it's funny. We goof around a lot on the podcast. So it keeps it entertaining. I think that's part of the reason why a lot of people listen to it, is that we keep it nice and fun.

**[00:54:30] JM:** Right. So I think this is a distinction between your show and mine, is that you're practitioner. You are actually working with this stuff and you have a deep understanding of how React works, whereas I am typically doing more of a survey of these kinds of things. But this approach to doing a tutorial style podcast, what's the key to explaining complex topics over audio? Because I've struggled with exactly what is the – Because if somebody is like in their car, or they're at the gym, to what degree can they process a tutorial if they're multitasking? Because if I'm watching your course on my computer, I'm going to be fully focused and I'm going to have a window open that's the terminal. Even then, I might have to rewind a little bit

and go back and be like, “Okay. Here's this thing.” What's the key to explaining complex topics over audio?

**[00:55:23] WB:** We make sure that or we try not to get into the syntax too much and more focused on core ideas. So we had an episode on React Hooks. So we would explain what are some of the problems with the way React works right now? What are some examples where you might use a class? Then what are React Hooks? What problems do they solve? What are some actual use cases where you'd want to reach for these things and how can you build your own?

Somebody going to work, riding their bike, driving their car, I like to think of it as surface area. They are now aware of React Hooks. They’re aware of the core concepts behind them, and then they can pick up one of my tutorials or go read the docs or something like that to actually see how it works. So it's more just about like people don't necessarily know that these things exists, or what they are or just kind of like – I like the word surface area a lot, is that sometimes you just like want to like tap into two guys talking about web development so you know what is out there and in which direction you should be going. People say all time, “I discovered Gatsby on the podcasts, and now I’ve  rebuild my website. Thanks so much.” How would somebody have discovered that if they're not like us and on Twitter 24/7?

**[00:56:38] JM:** Why have podcast grown in popularity recently?

**[00:56:41] WB:** I think it's because it's downtime. You can fill in your downtime by listening to a podcast at the gym. I listen to podcast when I'm going grocery shopping, when I'm driving my car. It’s just a great way to – Instead of watching a video tutorial or reading a book, is you have to like set-aside for that. This is sort of just passive learning, where if you're going to – Instead of listening to music or just the dumb radio station, why not throw something that’s helpful and forward your career a little bit?

**[00:57:11] JM:** What are the problems with the podcast infrastructure? Because whenever I look at the podcast infrastructure I’m like, “There’s something wrong here. Why does it work this way? Why is there no really good searchability over individual podcast episodes?” for example. I mean, I've been tempted to, “Maybe I can build something around podcasting,” but there’s such

a big graveyard of businesses that people have tried to build around podcast infrastructure. What are the problems that you see with the podcast infrastructure and do you think there's any business that could be built there?

**[00:57:43] WB:** I think searchability is probably the biggest one that we have. People ask us all the time, “When did you talk about this?” or “You should talk about this.”

I think like once a podcast becomes more than six months old, it's dusty and it's very hard for people to actually surface those. So we've been talking about how do we change our website to surface different ideas or top episodes or things like that? That’s probably a big one. The advertising model is another big one. I know that I listen to podcasts that I know are from like NPR and I get Canadian ads in them. I know that they are doing something in that process to geo-locate my IP when I download it and download the version with the Canadian ads in it. I think that is also kind of interesting as well.

We’re lucky, because our sponsors don't necessarily care about where people live just because as long as they're there web developers. But I know a lot of other podcast have that problem where you need to build to localize the advertising.

**[00:58:41] JM:** Right. So you built a very successful business teaching people and doing podcasting and doing teaching, and I guess in-person teaching. Do you have any desire to do a startup?

**[00:58:57] WB:** No, I don’t. I've always been interested in those things, but my main motivation in everything that I do is just to make a comfortable balanced family life that I have and being able to just like – Like I said, sit down and do whatever I want. I just want to go up to my office. I don't have that meetings and things like that. I just want to be able to build my own thing. So that's kind of what I'm doing, and I also just like absolutely love teaching people things. That why I got into that. So there hasn't been much of like, “Oh! That would be a cool startup,” or something like that. There're a couple of things around delivering video better, which I think could be fixed. I know there’re a couple of startups in that area right now, but it’s never something I’ve wanted to solve myself.

**[00:59:45] JM:** Okay. Well, Wes Bos, I want to thank you for coming on the shows. It's been really fun talking to you, and I'm impressed by your ability to know what you want and to get it. You've gotten it. You're there.

**[00:59:57] WB:** Yeah, totally. Well, thank you so much. I appreciate it. It's been fun coming on.

**[01:00:00] JM:** Yeah, continued success.

**[01:00:01] WB:** Thank you. I appreciate it.

[END OF INTERVIEW]

**[01:00:06] JM:** Continuous delivery frees up your developers to ship code independently of one another. GoCD is an open source continuous delivery tool, and we have partnered with GoCD to deliver some quick tips about modern continuous delivery. Today's tip is to have a governance policy around secrets. Secrets like API keys, passwords and certificates need to be accessed securely and you want to have a policy in place across your organization for how secrets are managed so that your continuous delivery pipeline doesn't become a security vulnerability.

To find out more about continuous delivery, check out gocd.org/sedaily. GoCD has been a sponsor of Software Engineering Daily for more than two years. We've done lot of shows with the GoCD staff and we've learned a lot about continuous delivery through those interviews. So we're proud to have their support. If you're getting started with continuous delivery yourself, check out gocd.org/sedaily.

[END]