CTRL+CLICK CAST #99 - Web Dev Trends with Laurie Voss

CTRL+CLICK CAST is proud to provide transcripts for our audience members who prefer text-based content. However, our episodes are designed for an audio experience, which includes emotion and emphasis that don't always translate to our transcripts. Additionally, our transcripts are generated by human transcribers and may contain errors. If you require clarification, please listen to the audio.

Preview: The only advice that I would have is don't wait too long to get onto the trend. If it's happening, it is happening and you should get on it. Otherwise, you're going to be left behind. But also don't wait too long to get off again. Nothing lasts forever.

[Music]

Lea Alcantara: From Bright Umbrella, this is CTRL+CLICK CAST! We inspect the web for you! Today we have Laurie Voss on the show to talk about trends in web development. I'm your host, Lea Alcantara, and I'm joined by my fab co-host:

Emily Lewis: Emily Lewis!

Lea Alcantara: Today's episode is sponsored by Foster Made, a versatile web development agency specializing in custom application development, content management systems and user experience design. Through partnerships with designers, agencies and organizations, Foster Made is committed to building better digital experiences. Visit fostermade.co to learn more.

This episode is also sponsored by Vector Media Group. Vector Media Group is a full-service, interactive agency based New York. They specialize in web and app design and development and online marketing. In addition to custom platforms, Vector is widely known as a top Craft CMS and ExpressionEngine partner. Get in touch today at vectormediagroup.com.

[Music ends]

Emily Lewis: Our guest, Laurie Voss, has been a web developer for 21 years and currently finds himself COO of npm, Inc., which he co-founded. Laurie cares passionately about the web and believes that everyone should be able to participate in making it better. He joins us today to talk about the rise of frameworks, component architectures, graceful degradation and other trends in web development. Welcome to the show, Laurie!

Laurie Voss: Hi, thanks for having me.

Lea Alcantara: So Laurie, can you tell our listeners a bit more about yourself?

Laurie Voss: Well, I grew up in Trinidad and Tobago in the Caribbean.

Lea Alcantara: Hmm.

Laurie Voss: And that sort of flavored my experience with web development and technology ever since. I've sort of approached all of my career as remembering what it was like when internet access was very hard to get and computers were so very hard to get.

Emily Lewis: [Agrees]

Lea Alcantara: Oh.

Emily Lewis: So if you grew up in that area in an internet access was very difficult, how did you get started working on the web?

Laurie Voss: Let's see, I think it was Christmas of 1996, my mom bought a modem through me begging and begging and begging, and I dialed up to a local ISP and we were measuring our download speeds in like bytes per second at the time or anything.

Emily Lewis: [Laughs]

Lea Alcantara: [Laughs]

Laurie Voss: Loading a web page is like you'd click a link and you'd wait five minutes for this thing to show up.

Emily Lewis: [Laughs]

Laurie Voss: But I remember being incredibly struck by the fact that like websites that Trinidadians had put together looked exactly as professional back in 1996 as the New York Times like the HTML-only had so many features.

Lea Alcantara: [Agrees]

Laurie Voss: There was no CSS, there was no JavaScript. So you could learn all of web development like all that there was to know in an afternoon and you can make something that look exactly as good as the New York Times could make also in the afternoon, and that was really a breakthrough moment for me. It was, "Oh, right, there's no barrier ahead with this internet. The fact that I'm very, very far away or rocking in the middle of the ocean, that's not a problem anymore."

Emily Lewis: That's cool. It reminds me of the first time I wrote my very first line of HTML, I was like, "Oh, wow, you can publish something instantly." It's kind of magical, a very new way of thinking.

Laurie Voss: Absolutely.

Emily Lewis: So today we're going to be talking about web development trends. Before we dive into some specific ones, Laurie, I'm curious how do you define a trend in web development?

Laurie Voss: Well, you told us that we would be talking about trends, and so I started thinking about it a little bit, and I think the metric that I settled on was growth.

Emily Lewis: [Agrees]

Laurie Voss: It's not about popularity, and it's not about any other metric, but it's about something that is getting very big very fast, like it's something that it will show thought leaders talking about this thing and like there will be conferences about it and tutorials and demos and every podcasts about it will be about it for six months and that's how you tell something to be a trend. But I think the thing about them is that they're often ahead of the curb, like the trend is not a thing that everyone is doing. The thing that everyone is doing is the last trend, but a trend is the thing that people are about to do.

Lea Alcantara: Ah.

Emily Lewis: Right. That makes sense. So let's see, if you don't mind, because this is putting you on the spot. We didn't send this to you in advance, but I'm curious when Ethan Marcotte coined the term "responsive web design," would you have considered that a trend at the time?

Laurie Voss: When was that?

Emily Lewis: Oh, it's about five years ago or so. About the time frame when it sort of took off, was it a trend? Would that be how you would describe a trend?

Laurie Voss: I would say the first time I heard the term "responsive" to refer to a web page was probably around 2012.

Emily Lewis: [Agrees]

Laurie Voss: So five years is about right, and yeah, I would say that's definitely a trend at the time. But of course, like a lot of trends, it's gone from being a trend to just being a best practice.

Emily Lewis: [Agrees]

Laurie Voss: You wouldn't build a website that's not responsive anymore, at least the hell you wouldn't.

Emily Lewis: So trends aren't fly by night necessarily. They can become established practices.

Laurie Voss: Yeah, and the best ones do.

Emily Lewis: Yeah. I think that's a good point. So let's talk about some of those trends. [Laughs] This first one, this is probably one of my more favorite topics these days, frameworks. So it seems like there's like a framework for just about every aspect of building website, designing, developing, everything really. What do you think are the benefits of frameworks today?

Laurie Voss: I think that people think of frameworks as having some kind of technical advantage. Certainly, the frameworks markets themselves that way, and there are sometimes limited technical advantages to using one framework over another, but I think the real advantage to a framework is that there is a framework. Instead of building something bespoke, instead of building a snowflake of an architecture that nobody understands but you, you're building something that somebody else has put together. So it's something that somebody else has presumably used to build something bigger than your start-up projects so they're running the edge cases and they're running to complicated things before you have. So if you follow the rules of the framework that were laid down to avoid that, you should be able to grow faster with them while hitting fewer roadblocks.

Emily Lewis: [Agrees]

Laurie Voss: And other advantage of a framework is that other people are using this thing, so you can reuse stuff that other people have written. You can read blog posts about it. You can ask Stack Overflow questions about it. Because other people are using it, it becomes more useful.

Emily Lewis: [Agrees]

Laurie Voss: So when I go to a startup and people say that they wrote their own framework, I'm like, "No, that's kind of a contradiction in terms, like it's not a framework unless lots of people are using it."

Emily Lewis: [Agrees]

Laurie Voss: If you're the only person using it, it's not a framework. It's just an architecture that you wrote.

Lea Alcantara: Oh, I find it a really good distinction to make it the difference between internal and external architecture as you mentioned, so are you suggesting that if moving forward, people should be using the term "framework" to suggest anyone can use it, not just an internal team?

Laurie Voss: I think the more people who use it, the more valuable a framework becomes.

Emily Lewis: [Agrees]

Laurie Voss: The more people are using it, the less time you have to spend explaining stuff, like some of the granddaddy of web frameworks, I think, is probably Rails. Rails revolutionized the web industry and think about Rails, it's not a very good framework, like it's kind of weird. [Laughs]

Emily Lewis: [Laughs]

Lea Alcantara: [Laughs]

Laurie Voss: And you know, there's a bunch of monkey patching going on and it's not super easy to understand, but the thing about it was that they were like, "Do it this way," and suddenly there was a huge pool of people who knew how to build a Ruby on Rails application.

Emily Lewis: [Agrees]

Lea Alcantara: Right.

Laurie Voss: So I remember back in 2003 or whenever, if you've hired somebody to work at your company, the first three months that they were working there was you just explaining to them your architecture like, "This is where we put our HTML. This is where the SQL goes. This is where our database clones are," and like ramping up on that architecture was as slow and laborious process from building to the hiring, and Rails completely changed that because you could hire a Rails developer onto a Rails app and they could start on the first day. They'd be like, "I know where everything is. What is it that you want done?"

Emily Lewis: [Agrees]

Lea Alcantara: [Agrees]

Laurie Voss: And so that's where I think the advantages of frameworks is. It's in the people. It's in the less time spent learning things.

Emily Lewis: Yeah, absolutely. I was here in Albuquerque; they just started a new program. They're like 10-week coding boot camps, fulltime coding boot camps, and they have a full Stack developer/programmer boot camp, however you want to call it, and they're focusing on using frameworks to teach, and the students are students who have no technical, I mean, they may have some technical background, but no coding background, they've come from other industries. So it's a way of kind of giving them a very "quick" way to learn how to build something within that 10-week program. Also, to your point, Laurie, some of the frameworks that they have incorporated into the curriculum are specifically because the city is working with this program and the city needs people with that framework knowledge. So for the exact reason that you explained, when someone is hired and it's on-boarding, they already have that fundamental framework understanding.

Laurie Voss: Exactly.

Lea Alcantara: So I wanted to take a little bit of a step back. You mentioned about frameworks being very technical, so it's clear that frameworks are development related, right?

Laurie Voss: [Agrees]

Lea Alcantara: But do you think frameworks are better suited for development? How about design, like UI frameworks?

Laurie Voss: I think they have the same benefits there. I think if a lot of people have established that this is the way that you design things and you can onboard a designer faster the same with even onboarded a developer faster, I think depending what kind of a design framework and how prescriptive it is, it can become a problem.

Timestamp: 00:10:11

Lea Alcantara: [Agrees]

Laurie Voss: So if everybody's code looks the same, that's an advantage because only the developers will ever look at it. If everybody's website looks the same, that gets kind of dull. Bootstrap is really, really popular as a way of just getting things started for the CSS one project for a while, and there was definitely this period where you were like, "Oh God, it's another Bootstrap website. These things all look exactly the same."

Emily Lewis: [Laughs]

Laurie Voss: But on the other hand, visual consistency is a benefit. Like people expect that the thing on the top left is a logo that you can click and it will take you home, like those sorts of universal consistencies of design make everyone's experience of the web faster. So it's not a homerun as it is I think with server-side frameworks, but there's definitely a place for them in design and everything else.

Emily Lewis: [Agrees]

Laurie Voss: Like what does accept? It's like framework for running your team.

Emily Lewis: Yeah, that's true.

Lea Alcantara: Right.

Emily Lewis: Internally at Bright Umbrella, we have called it a framework, but now that I've heard your points, I think we would just call it our own custom architecture because I'm not a fan of frameworks personally for the kind of work that I do, and so we created our own, essentially, architecture set of rules for our front-end development. I could even call it like a pattern library or something like that for our HTML and CSS patterns that I use so that when we start a project, I can get the project up and running a lot faster and kind of leverage the benefits of a framework where you can get something started quicker, but have less of the challenges of frameworks, which kind of leads me to the question of, what do you think the challenges of frameworks are, some of the cons of relying on a framework?

Laurie Voss: I think the problem with a framework is when it begins to become too specific. If a framework has a lot of opinions, if it's decided that this is the way that you should build things, the chance is that the decisions that they've made, the sort of space that they've mapped out for you fits your problem perfectly gets smaller and smaller.

Lea Alcantara: [Agrees]

Laurie Voss: So I forgot the name of a startup a couple of years ago that built this amazing, amazing web app that let you build an entire web app in a GUI in the web so you can use a web app to build another web app.

Emily Lewis: [Agrees]

Laurie Voss: And their demo was this amazing demo where you could put together basically like a Twitter clone in about 15 minutes. But the thing was, the company failed for this reason was that it was basically all you could build. It was too specific. It was too designed for one set of imagined use cases, and if you want or need to do anything complicated, you need to break out of the framework.

Emily Lewis: [Agrees]

Laurie Voss: So sitting at npm, we have an enormous amount of data about what frameworks people are, in fact, using, and so I want digging through our data today to take those out and find out what they were. So the top five, and the top one by a long way is React, which is not really a framework, right?

Emily Lewis: [Agrees]

Laurie Voss: It's like a framework, but it's not all of the framework and I feel that's part of why it's so popular is that it's not all of the framework. It's just solving one problem really, really well. And number two is jQuery, which jQuery is not a framework at all, right?

Emily Lewis: [Agrees]

Laurie Voss: You don't think of it as being a framework at all, and it's incredibly popular. It's not about trendy. It's just massively, massively popular, like jQuery is not even primarily installed at npm. So if you include all the people who are bringing it in via script tags, it's probably still by far the most popular framework, if you want to call it that. Then there's Angular and then there's Backbone, and then there's Ember and then there's Vue.

Emily Lewis: [Agrees]

Laurie Voss: And I think that if you sort of look at that list and so go down the list, you'll see that they're getting or going from less to more specific.

Emily Lewis: [Agrees]

Lea Alcantara: [Agrees]

Laurie Voss: The further down, the more specific it gets and the less popular the framework becomes, and I think that's because when it gets incredibly specific, it becomes harder to map your use case to it.

Emily Lewis: Absolutely, yeah, that makes sense. It also kind of highlights another challenge of frameworks I think, which is the frameworks that are so broad, they do not address the edge cases, and frankly, this is my concern more from an education, a new developer kind of perspective. If a new developer has, let's say, taught themselves Bootstrap and then their client or employer has a need that Bootstrap can't meet, I have concerns that they can then solve those problems on their own without the framework support, that they still have the fundamental skills to solve design or development problems without the kind of community brainpower that put together the framework in the first place.

Laurie Voss: I think I'd probably take a different position on fundamentals to you if only because after 21 years in the business, I'm not sure what people think out as being fundamental anymore.

Lea Alcantara: [Agrees]

Emily Lewis: [Agrees]

Laurie Voss: Like when I started web development, the first book that I ever bought to teach me web development ... The very first thing that the book taught me was CGI.

Emily Lewis: [Laughs]

Laurie Voss: They were like, "This is the CGI…"

Lea Alcantara: What? [Laughs]

Laurie Voss: "This is the CGI protocol. This is how you write a PC software that knows how to speak HTTP," because there were no frameworks. There was nothing. There was nothing that knew how to speak HTTP out right off the bat, and so it was like, "We assume that you're writing either Perl or C," because this is 1995 and PHP hasn't been developed yet and neither has JavaScript, "and you're going to have to write from scratch a web server," like they did not assume the existence of Apache, far less Nginx or any of the rest of them.

Emily Lewis: [Agrees]

Laurie Voss: And they were like, "So these are the fundamentals, here you go," and instead of teaching you HTML, the book taught you SGML, which most people don't even remember anymore.

Emily Lewis: [Agrees]

Laurie Voss: SGML was like this meta language that HTML is just one application of, and so [laughs], when I sit here and people are talking about, "Oh, you've got to get to the fundamental of HTML," I'm like, "No, HTML is a framework. All computers are frameworks on top of frameworks on top of frameworks."

Emily Lewis: Oh.

Laurie Voss: Everything is an abstraction on top of another abstraction, and the stack is so big and so complicated that it's impossible for anybody to know every layer of the stack.

Emily Lewis: [Agrees]

Laurie Voss: The idea that there's a full-stack developer is a myth.

Emily Lewis: [Agrees]

Laurie Voss: Like the stack have 45 levels. If you've got ten of them, you're doing really well, but you're not a full stack developer, and nobody is, so if the very first thing somebody learns is how to build an app in React and Redux, and that's all that they know how to do, they're fine. They can build 95% of web apps. And if they need to do something complicated that that thing can't do, then they'll figure it out, they'll break out of the abstraction eventually. But I think I can't define what a fundamental is when it comes to web development, and therefore, it probably isn't a real thing.

Emily Lewis: I have never thought of that that way, but it really make sense when you explained it that way, even just conceptualizing HTML as a framework make sense. It's very interesting. So I don't want the whole episode to be about frameworks. [Laughs] So let's talk a little bit about some of those component architectures like React that you had mentioned. How do you see the difference between, let's say, React and a framework? What makes it different? Is it because it is more narrowly focused?

Laurie Voss: Yeah, absolutely. Framework is usually or often is sort of soup to nuts idea. It's like, "This is how you'll start this. This is how you'll end. This is how all of the things that you need to do to put an application together will work inside of this framework." And React takes a very different view. React says, "This is how you lay out your HTML, this is how you will respond to events, and this is how your views will re-render themselves when data changes." But it has almost no opinions about anything else. It has no opinion whatsoever about routing, which is kind of crucial if you're building a web app of any complexity at all, the data layer, it has a sort of menu of things that you can use. Redux is quite popular, but it's not running away with it by any means, and I think that's part of why React is so overwhelmingly popular. It's a really good size of abstraction. It takes this one thing that's very easily moved into anything else and it absolutely nails it, like people are using React both to do web apps and also desktop applications and also mobile applications. The same framework is running in all three on my desktop right now. [Laughs]

Emily Lewis: [Agrees]

Laurie Voss: There are applications running that are written in React as well as in my browser, and that is an amazing success as a framework does, and it's because it was so constrained in what it chose to address.

Lea Alcantara: So it feels like we're even talking a lot about the pros, and even though Emily asked about the cons about particular frameworks, I'm not sure we really dove as deep as it could have gone, because, for example, you mentioned Rails as pretty awful. [Laughs] So wouldn't that be a con, like in regards to, well, now, that there's this massive adoption of these things and yes, people can go from soup to nuts to this, why would you say that, "Well, this particular framework is awful," or what is awful about certain frameworks?

Laurie Voss: First, I think I don't want to bash Rails. I wouldn't say that it's awful.

Lea Alcantara: Yeah. [Laughs]

Laurie Voss: I would just say that we have built things since then that are better. The challenges for framework are if you pick the wrong one, then you end up in this very small pool of people. The advantage of a framework is that if everybody is using it, all of the problems are solved for you by other people. If you've picked the wrong one, if you've picked one that's declining in popularity, then you're going to end up with a pool of people who are less and less able to help you solve your problem.

Emily Lewis: [Agrees]

Lea Alcantara: [Agrees]

Laurie Voss: And you're married to this framework now because you've written everything in this framework and you'd have to get out of it. You have to do a rewrite. But if you've not used a framework, if you've just written entirely your own thing, that would also be the case, the pool of people would be help you is one, it's just yourself. So some frameworks are better than others, but I think in general, if you pick a pattern and you share it with other people, that's a universal good. It's never going to be the situation where you're doing it entirely yourself from scratch is going to be better than picking something that at least one other person uses.

Timestamp: 00:20:16

Emily Lewis: So to continue a little bit on the discussion about challenges or cons, and this is more specific to like a front-end framework like a Bootstrap or something like that, I find it challenging. I guess… [laughs] perhaps this is too much of a reflection on my desire to be a control freak, but I don't always care for the way that someone else decided a pattern should be written or how that HTML should be structured, and so I mean, that's why I choose not to work with frameworks.

But we inherit projects that another developer started with say Bootstrap and one of the challenges we found is this one site we inherited that was built with Bootstrap in 2011, it's so tightly coupled to whatever was available at that time. To try and uncouple it and do new things becomes very time consuming, kind of a rabbit hole challenge with this particular site, and so it increases costs for the client as well in terms of we can't scrap it, we have to work with it, but it almost doubles our time for certain projects. So that's one challenge.

But the converse, not the converse, but another side of it is I was watching a thread on Twitter. I think it was yesterday. It might have been this weekend, but people were kind of talking about frameworks and why some people choose them, some people don't, and one person's comment is one that I relate to myself, Sarah Souedin was tweeting that she would rather write her own from scratch than try and fix what someone else built in a framework. For example, like if there's some sort of accessibility needs that weren't addressed by, say, the framework, then retrofitting it to get it to work to meet some sort of accessibility compliance would be more challenging than writing from scratch. I think that's a relevant scenario. I think it's one of those things that it's kind of before you dive into development, you need to really think about those sorts of things before you decide you're going to rely on a framework or some sort of library.

Laurie Voss: Yeah, I think one of the ways that one can sort of go down the wrong path using a framework is you can adopt it when what you're doing is too simple to really require it.

Emily Lewis: [Agrees]

Lea Alcantara: [Agrees]

Laurie Voss: Like people are long time web developers who spent ten years rolling their eyes at people who are using jQuery to put together a single page form that had one email capture box on it.

Emily Lewis: [Laughs]

Laurie Voss: And you're like, "Well, why is jQuery on this website? This form has one box and one submit button. What is the JavaScript for?"

Emily Lewis: [Laughs]

Laurie Voss: And that can often be the case. If you're putting together a single page, relatively static marketing site, why are you doing this in React? You don't have components here using each one of these boxes exactly once, what is the reusability? What is this for? So yeah, just reflexively jumping into a framework because it's the only way you know how to throw something together can backfire sometimes. But if you're talking about stuff like accessibility and things like that, if the framework you're using isn't accessible, pick a framework that is or improve the framework, right?

Lea Alcantara: [Agrees]

Laurie Voss: Because then you don't just get it on that project. You get it on every projects after that, and most of the people who've put together these frameworks, they're pretty receptive to that kind of stuff.

Emily Lewis: [Agrees]

Laurie Voss: They're like, "Yes, we realize that that was one of the things that we haven't built yet is ARIA compatibility for all of this stuff and how do we do that?" And you give them the tool, then they'll run with it, I think.

Emily Lewis: Oh, I love that suggestion. So do you feel that component libraries or architectures share the same pros and cons of frameworks or do you think that maybe they're more beneficial because they're a little more focused for needs?

Laurie Voss: I think it's important to drive a distinction between what kind of components we're talking about. Obviously, npm is a gigantic registry of 500,000 reusable components, and so just the sheer number of users that by date a half million users now, like that's a fairly successful component model one has to say, right?

Emily Lewis: [Agrees]

Laurie Voss: Not every single thing in there is getting reused, but lots of those things are getting used millions and millions of times a day, that's a component model that works, turning those to fit the models that could be exported. I don't think anyone would argue against that. On the front end, however, reusable components had been the holy grail for a long time, right? Like the closest we got for a while, I think, was jQuery plugins where we'd be like, "Oh, thank God, somebody has written this calendar widget for me and I can just drop it in there." But still half the time, it would look gross in CSS where it would interact now with your CSS or it would do something strange and you wouldn't be able to change it, and React is the closest I've seen to genuinely reusable component architecture on the front end and that I think is why React is absolutely running away with it.

Like I said, we sit at npm where we can see people uploading and downloading React components on the registry. It is where a lot of our growth is coming from. It is an enormously prolific and popular ecosystem growing much, much faster than any other ecosystem in web development right now, and it's because it looks like somebody may have finally cracked the code of how to do reusable front-end components and part of it is the way that they did that, which just broke all the rules. They're like the HTML is in there, but it's not HTML, it's JavaScript, and the JavaScript is right next to it, and the CSS in there too, and the CSS is also a JavaScript, and everyone was like, "What is this? This is breaking all of the rules. Our beautiful separation of concerns or 15 years of received wisdom in web development has been completely thrown out of the window by React, and the result is something much, much better than web development had come up with in the past 15 years." [Laughs]

Emily Lewis: [Laughs]

Lea Alcantara: [Laughs]

Laurie Voss: So yeah, I'm completely on board with component architectures. I think they work well and I think React, in particular, is tremendously exciting. It's also a problem I've never seen solved properly before.

Emily Lewis: With this new problem being solved, what do you see happening? Do you see anything new with more innovation coming from what React has introduced to sort of breaking the rules mentality, what else might come from it?

Laurie Voss: Hmm, that's a really excellent question because what you're saying is, what is the thing that we weren't doing because we were consistently redeveloping our UI, right?

Emily Lewis: [Agrees]

Laurie Voss: Like if you're putting together a web app and you say you were building it on top of Rails and wanted to do the back end in frameworks, you're still five years ago starting from scratch with a blank sheet of paper for your CSS and your HTML and working out what it's going to look like, and if you've got reusable components, what is it that you can do? I think the answer is probably more complicated web apps than we've ever seen.

Emily Lewis: [Agrees]

Laurie Voss: I think we will be able to build or spend more time building features and less time getting the individual behavior of one component to work right and I think that's part of why it's turning up in desktop application design, right?

Emily Lewis: [Agrees]

Laurie Voss: Because web application's simplicity is kind of a virtue. You want it to look well on a phone, and phones don't have a lot of space for buttons in the UX and stuff like that, but desktop applications are desktop applications, you can kind of put in there that you want, and the fact that you can use a technology that was built for the web to desktop things is interesting.

Emily Lewis: [Agrees]

Laurie Voss: So I think it would mean that probably in the medium term that mobile app development, desktop app development and web development will begin to move closer together.

Emily Lewis: [Agrees]

Lea Alcantara: [Agrees]

Laurie Voss: I think that's already beginning to happen and I think that trend is probably going to continue.

Lea Alcantara: Well, one thing is that is a little bit concerning in regards to all this. I mean, this sounds super great like it sounds very advanced, but our listener, Christopher Kennedy, specifically wanted you to be on our show to get your opinion on graceful degradation.

Emily Lewis: [Laughs]

Laurie Voss: [Laughs]

Lea Alcantara: So we're talking about all these fancy features and things like that that often needs to have, every bell and whistle in terms of your browser and JavaScript and all that fun stuff. So what is your opinion on graceful degradation? Is that even a trend itself, too?

Laurie Voss: Yeah, graceful degradation is a trend of the trends that was trending ten years ago and then it lost favor and then sort of came back.

Emily Lewis: [Agrees]

Lea Alcantara: [Agrees]

Laurie Voss: I wrote a blog post about this, I don't know, about a year ago or two years ago now after seeing a bunch of very, very good web developers who came to a huge trigger fight about this.

Lea Alcantara: [Laughs]

Laurie Voss: There are basically two things that people think of as graceful degradation. There is what web apps do, which is sort of the offline first movements like, "We will load your entire web app and then if you lose connection, everything still works, and when your connection comes back, all the actions that you've been taking, they sync up to the server in the background." The thing that's degrading there is the network connection, and for a web app, that works fine because you get exactly one chance to load a web app and then it sits in your browser, it sits on your phone until you're done using it. But if you're using an app for an hour, the chances of you losing internet connectivity in that hour, especially if you're on a mobile device is quite high.

Lea Alcantara: [Agrees]

Laurie Voss: So being able to withstand that kind of degradation, that makes a lot of sense. One of the problems with that is that it means that you have to preload a bunch of stuff and that makes it slower to load in the first place, and that's where the other set of people begin to yell because if you think about a website, if you're used to be using what I would call a "flat" website like a media website, New York Times or something like that, people don't load an article on the New York Times and then use it for an hour.

Emily Lewis: [Agrees]

Lea Alcantara: [Agrees]

Timestamp: 00:29:51

Laurie Voss: They need it to load as fast as they possibly can, and also they'll do it dozens of times a session if you're reading lots of articles, so instead of this one chance to fail to load, you have lots and lots and lots of chances to fail to load and then the network is still flaky so the chances one of those things failing is much higher. So that's when you get to progressive rendering, which is the other kind of graceful degradation. It's the idea that you don't need all of the pieces of your website to load for your user to be able to do something useful.

Emily Lewis: [Agrees]

Laurie Voss: You want to load the HTML as fast as possible, render it as fast as possible and then get the styles in there, and then get the JavaScript in there, and if your connection is interrupted at any point in that one second process, you still got something that works. But if what you've got is a web app, that's never going to work, right?

Emily Lewis: Right.

Laurie Voss: If Gmail has only 50% of Gmail loaded, Gmail isn't useful and there's nothing you can do about that. But if the New York Times article loads only the text and not the images, 90% of the utility of that article is still there. So I think people are sort of getting to a fight about graceful degradation because there are two of them and you have to tailor what use case you're optimizing for.

Emily Lewis: Yeah. I think that's a really good distinction, particularly now that we're dealing with so much and conversation is just so much different today than it was ten years ago when I think it was a bit of a trend to talk about progressive enhancement or graceful degradation. But when you are talking about apps, it is a completely different discussion and I think it's important to make that distinction. I am curious though. These topics within the web development world, are they popular topics? Is it something that you see other developers struggling to define or struggling to approach in a user-focused way?

Laurie Voss: Oh, that's an interesting question. I think those sorts of things are the topics that really high power web developers working at large corporations where you have big teams putting together complicated websites. They're the sort of things that they care about. There are also those sorts of things we talk about in conferences or talk at conferences all the time.

Emily Lewis: [Agrees]

Laurie Voss: I think the average web developer is just trying to get something out of the door by Thursday, right? [Laughs]

Emily Lewis: [Laughs]

Lea Alcantara: [Agrees] [Laughs]

Laurie Voss: They are not so interested in whether it works offline or whether it progressively renders, as long as it renders at all, because it needs to be out by Thursday. So there's a very long tail of developers I think that are making this stuff. The time required to get that stuff right will always be a luxury.

Emily Lewis: Yeah. I think that is frustrating because it is something that, regardless of what you're building, you do want it to be accessible to whomever is trying to access it, and yet, we are in this world of "it's due on Thursday." [Laughs]

Laurie Voss: Accessibility is its own driver. If enough people are doing something, best practice is changed to adapt to that thing.

Emily Lewis: Yes.

Laurie Voss: I think where that's most obvious is the trend for mobile web access with something like three or four years ago, Google flipped over to being 50% of all of its traffic is coming from mobile devices. That's a tremendous shift, right?

Emily Lewis: [Agrees]

Lea Alcantara: [Agrees]

Laurie Voss: In the 2000s, there was this brief period where everybody had giant screens and fast connections and you could be absolutely positive that everyone had a lot of screen real estate to look at your web app and they were running it on a fast enough computer, that processor wasn't a problem. It was all Shangri-La for about ten minutes. And then in 2007, Apple introduced the iPhone and suddenly, everybody was browsing the web on tiny, tiny, little screens that could barely run JavaScript on extremely limited processors, and it took a couple of years for people to realize that it had completely changed the landscape. Now, everyone is doing responsive web design because we have no idea how big our user screen is anymore, not even to a vague approximation, and websites are just getting simpler because so many people access them on phones where there's just not a lot of room to cram in a lot of complicated stuff.

Emily Lewis: [Agrees]

Laurie Voss: So you should begin to see where things like hamburger menus on desktop website, and I'm like, "Why is there a hamburger menu there?"

Emily Lewis: [Agrees]

Laurie Voss: You have plenty of desktop real estate, and it's because very few people access that website on a desktop, and most people access it on the phone and a hamburger website makes sense there, and we have to get it out of the door by Thursday, so we didn't bother to come up with a desktop view for this primarily mobile website, and that is a really fundamental shift.

Emily Lewis: I think that's good point out. I think it will be really fascinating to see how things continue to evolve in terms of what people are using, and then how that in turn affects what our "best practices" are.

Laurie Voss: Yeah, absolutely. I was just looking at some stats this morning. Three years ago, 5% of the traffic to npm's website was from mobile devices, and this month, it's 10%, and 10% is not a huge amount, but I think that's amazing when you consider there's almost nothing that you do on our website, that makes sense on a mobile device, right?

Emily Lewis: [Agrees]

Laurie Voss: You could, I suppose, manage your teams and stuff like that from a website, but I can't imagine a lot of people are doing that, and I asked people around what is it they're doing on npm's website on a phone, because it's one in ten of you, they've got to be doing something important.

Emily Lewis: [Laughs]

Laurie Voss: And it turns out that the use case is people who code primarily on a laptop where they've gotten limited screen real estate and they want to keep the documentation open for the module that they're reading.

Emily Lewis: Oh.

Lea Alcantara: Yeah.

Laurie Voss: You've done this.

Emily Lewis: Yes. [Laughs]

Lea Alcantara: I have done this. [Laughs]

Laurie Voss: Right. That's what it is. It's people reading the documentation on their phone while they're coding on a laptop, which I think is amazing. [Laughs]

Emily Lewis: I love that.

Laurie Voss: That's 10% of our views now.

Emily Lewis: I'm curious, how did you figure it out, just asking people how they were using it? How did you figure out that that was the scenario?

Laurie Voss: Yeah, absolutely. I have the advantage of I routinely give a talk at it's called Stuff Everybody Knows, my various web incubators. I'm meeting novice web developers in groups of 200 at the time on a fairly regular basis.

Lea Alcantara: Wow!

Laurie Voss: And it's fascinating. It is amazing talking to people who are just going to the industry and hearing the things that they think of as fundamental and always true and the things that they think of complicated and strange. These are people who they grew up in React and so they're like, "Well, obviously, that's where you put this date and thing is a prop, and we do this other…" and like they're ES6 features that I'm only just getting used to, and then you're like, "Oh, `document.getElementById()`," and they're like, "What's that? They're complicated." [Laughs]

Emily Lewis: [Laughs]

Lea Alcantara: [Laughs]

Laurie Voss: And it's just like the things that they think of as being easy and the things that they think of as being hard are very different.

Emily Lewis: [Agrees]

Laurie Voss: Which is why again when people talk about fundamentals, I'm like, "Five years from now, the people who grew up in React is going to be still talking and complaining that people don't know the fundamentals of React because we've built some framework on top of React."

Emily Lewis: Yeah. Oh, I love this. That must just gave you an incredible perspective on the industry and where things are going just to have the experience so often with new developers.

Laurie Voss: It's absolutely fantastic. So if anybody listening would like me to come and talk at their coding boot camp, please invite me because I love doing it.

Emily Lewis: Well, you mentioned your Stuff Everybody Knows talk and I was actually checking it out myself, and one of the points you bring up in the talk is about performance, and I'm just curious, it's become an increase. You see it as a topic much, much more on digital magazines and blogs and Twitter and everything else, and then, of course, as mobile has become more important in Google's mobile-first push, performance is also more important, and so I'm wondering if you feel like performance itself is a trend or does it fall more in those sort of best practices/foundational things?

Laurie Voss: Like I was saying, I think it's an adaptation to the new landscape that we found ourselves in.

Emily Lewis: [Agrees]

Laurie Voss: The primary device that people access the internet on is a phone now, and the phones have, even in North American and Western Europe, phones are fairly noted. And as the number of people online grows, the percentage of people online who are accessing it on First World internet connections and latest model high-end devices get smaller and smaller with the percentage of all of the people who are accessing the internet, so performance is becoming more and more of important.

Lea Alcantara: [Agrees]

Laurie Voss: Because if you need to access or if you're going to address the one billion internet users in India, you need to understand that the way that they access the internet is very different from the way that we access the internet. They access it for five minutes a day. They download everything and then they play with it for a while. Facebook I think has done an amazing job of adapting to this new landscape. They sent teams of people to live in India for months at a time and used the internet as people in India use the internet, and they came back and said, "We cannot use this app. This app is completely useless. We have to change it entirely." Like Facebook Lite is a totally different experience because it was built with a totally different set of expectations and it's doing very well for that reason.

Emily Lewis: So I mean, in a way then, I guess it sounds like performance is a trend in the sense that it's a huge…

Lea Alcantara: It wasn't thought of as importantly as now.

Emily Lewis: Yeah, I mean, it's evolved, you know?

Lea Alcantara: Yeah.

Emily Lewis: Like you said, it's an adaptation, and in many ways, this trend sort of reflect that, pushing the boundary, pushing the curve, responding to something new that we didn't expect.

Laurie Voss: Yeah, the trend is mobile web access, and performance is the reaction to that trend in and of itself.

Lea Alcantara: [Agrees]

Emily Lewis: Oh, good, yes.

Lea Alcantara: Good distinction, yeah. Okay, so we've been talking a lot about all of these particular technologies that developers can use, but we haven't discussed why someone should use them and how to evaluate. So what do you feel what are the factors developers and development teams should consider when evaluating a trend to implement, like let's say React or whether to learn it?

Laurie Voss: It's really tricky because it's hard to tell off the bat. The way to tell a trend is really going to work for you if everybody is already doing it. Like React, so many people are using it, it''s really not going to go anywhere. A really good example is MongoDB. MongoDB seven years ago was like the trendiest of databases, and I remember taking a look at MongoDB and its like actual performance characteristics and its failure modes and stuff like that, I'm just like absolutely table flipping. [Laughs] I'm like, "I'm never using this database. This database is a clown shoes database. I'm never going to do something with this."

Lea Alcantara: [Laughs]

Timestamp: 00:39:52

Laurie Voss: And this week, MongoDB filed for an IPO, and I was like, "What, like the joke database is filing for an IPO?" And I went looking into their website and I went digging into their performance characteristics and their failure modes and they've fixed all of those problems. They've had seven years. They just got critical momentum and critical mass and enough money that they could throw enough people that this is program is to fix every problem that they had, and it works, right?

Emily Lewis: [Agrees]

Laurie Voss: They're going to go IPO. They've actually solved a lot of problems for a lot of people. So to some degree, it doesn't really matter how good or bad the trend is as long as everybody is on board. Enough people will show up and fix whatever problems exist.

Emily Lewis: [Agrees]

Laurie Voss: But at the beginning, it's impossible to tell, right? You couldn't tell seven years ago that MongoDB was going to be big enough to go IPO. That's certainly not what I would bet. So if you're picking up a framework that's just getting started, I think, that not a lot of people are using, you just have to go with, "Does this make my life easier or now, and would I like it? Does it make me feel comfortable as a developer?"

Lea Alcantara: [Agrees]

Emily Lewis: And I think another question you could ask, and this was presented to us on Twitter by our listener and a former guest, Ben Furfie, "Does it solve your problem? If not, then don't use it."

Laurie Voss: Exactly.

Emily Lewis: And I think that makes perfect sense, especially like you said in the beginning when there's really kind of no way of knowing what the future holds, but you can know if you have a problem that whether it's going to solve it or not.

Laurie Voss: Exactly.

Emily Lewis: So when I introduced you at the start of the episode, Laurie, I mentioned that you think everyone should participate in making the web a better place. I'm wondering, is this idea of making the web a better place to sort of soft skill, is that something that should be a trend in our industry?

Laurie Voss: I think what I'm talking about when I'm talking about making the web better for everyone is making it possible for everyone to participate making the web better.

Emily Lewis: [Agrees]

Lea Alcantara: [Agrees]

Laurie Voss: And by making the web better, I usually would be making the web bigger. I think to what I was talking about in the first day. I said, "It was 1996 and I'm a 16-year-old on a rock in the middle of the ocean was looking at internet going, 'I can publish a web page today exactly as many people can see it as can see the New York Times, I'm going to do that. In fact, I'm going to do that for the rest of my life.'"

Emily Lewis: [Laughs]

Laurie Voss: I remember making a very clear decision, and it's 21 years ago now that I made that decision and I don't regret it. So I think the thing that I would encourage, if I can encourage anything, is welcoming people on board. Anything that makes web development more accessible to developers, anything that makes it easier to write web apps, simpler to write web apps faster to write web apps, anything that encourages people to participate who are currently being discouraged from participating, anything that makes more people get involved in making the web better, I'm on board.

Emily Lewis: [Agrees]

Laurie Voss: So that encompasses everything. That's like stop looking down at people who are using web frameworks or stop looking down at people who are using the language you don't like. Just stop looking down at people in general, right?

Emily Lewis: [Agrees]

Lea Alcantara: Right.

Laurie Voss: Like don't. Our industry is rife with stories of people who are prejudging other people in the basis of gender or race or economic background or language or whatever, and we just cut it out. That's the thing that I want to stop doing. I'm just saying, "Let everybody just build web pages."

Emily Lewis: [Agrees]

Laurie Voss: And the ones that are good web pages, other people will visit and ones that are bad web pages, other people will not.

Lea Alcantara: [Agrees]

Laurie Voss: And that's really the only metric that matters.

Emily Lewis: Yeah, I like that, and I think that that's something that I do wish to see that be a trend. Maybe trend is not the right word, but a popular topic in our industry. Even myself, I started in this field about 20 years ago as well, and especially when I got into the web standards movement, it was, "That was the only way to build, anything else was unacceptable," and it was just so rigid with my worldview.

Laurie Voss: [Laughs]

Emily Lewis: And as I've obviously matured and gotten older and worked with more people and taught more people, you get softer about your views and you see more reasons for why there are always exceptions to every rule, and I think that is something that I think as important to talk about as our mentality in how we think of our work and other people executing that work. It's just as important to talk about as the technology we're using. Like you said, looking down on people for using frameworks, I have been guilty of that, and it's more about questioning, "Does it work for you?" If it works for you, then understanding why it works for that person and maybe being open to how their view might influence me to not be so narrowly focused. That's something I personally would love to see have a greater focus on in the tech industry in terms of embracing the people part of it.

Laurie Voss: I completely agree.

Lea Alcantara: Right. And I mean, this just reminded me of something Laurie said earlier in the show in regards to frameworks where someone could reach the end of the utility of that framework, but that doesn't necessarily mean that person stops learning.

Emily Lewis: [Agrees]

Lea Alcantara: If they had the idea to learn the framework in the first place, they have the same brain to learn that what else is out there to fix whatever problem that that framework has stopped fixing, right?

Emily Lewis: [Agrees]

Laurie Voss: Exactly.

Emily Lewis: So Laurie, is this idea of making the web a better place a bigger place for people for people to participate? Is that something that inspired you to start the lgbtq.technology Slack?

Laurie Voss: It sort of become my hobby is starting online community spaces for queer people. It's funny because I keep doing it like this is the latest one, but I've done it like four or five times before with extremely varying degrees of success.

Emily Lewis: [Laughs]

Lea Alcantara: [Laughs]

Laurie Voss: I can't say that I have some like gigantic thesis going on with that one in general, like getting more people involved is great, but often, it's just an experiment, like lots of the time, I just wanted to try out a new web framework. I built a new form website using this new web framework because I couldn't think of anything else, I made it like gaygeeks.org, which I ran for a long time and people showed up and it became like a clubhouse for some very small group of people. Yeah, I can't say that I had some grand thesis. I'm really, really gratified by the response to it though. There's something like 3,000 people in there now from all over the place.

Emily Lewis: Wow!

Laurie Voss: And people have told me, "I've got jobs from this. I found friends from this, like it's made me feel less alone in the world." Like it's awesome, I can't say that that was what I was intending, although in fact, early on in the life of it, a bunch of people sat me down and said, "That thing that you were expecting this thing to be, it's not that. It's going to become this other thing instead," and the smartest thing that I've ever done with that community is go, "You're right. What you think it is is better than what I thought it was going to be. You should run with it."

Emily Lewis: [Agrees]

Laurie Voss: So I barely do anything with it anymore. It's run by a very, very selfless group of volunteer admins and really all that I do is make sure that Slack doesn't shut us down every so often. [Laughs]

Emily Lewis: [Laughs]

Lea Alcantara: [Laughs]

Emily Lewis: Just for our listeners who might be interested, can you describe the community a little bit?

Laurie Voss: So it is just a Slack community. I've seen most people are familiar with Slack at this point that welcomes queer people of any description who are involved in tech in any way, and it prioritizes the safety and comfort of marginalized groups over most other things.

Emily Lewis: [Agrees]

Laurie Voss: And that is an interesting choice to base a community around.

Emily Lewis: [Agrees]

Laurie Voss: So there are lots and lots and lots of channels, hundreds of channels for people to talk about all sorts of things, and it avoids them sort of normal failure modes of a community space where if you get 3,000 people in a room, there's absolutely no way that a fight isn't going to break out.

Emily Lewis: [Laughs]

Laurie Voss: So each channel has only a couple of dozen people in it, and as a result, those dozen people get along fine, and everybody is sort of in the meta community and they can bounce around, but everyone is not forced to bump into a bunch of people who have roundly different assumptions and life experiences to them, and that sort of works pretty well.

Lea Alcantara: So Laurie, do you have any final advice when it comes to getting on the dev trendwagon? [Laughs]

Laurie Voss: [Laughs]

Emily Lewis: [Laughs]

Laurie Voss: The only advice that I would have is don't wait too long to get onto the trend.

Emily Lewis: [Laughs]

Laurie Voss: If it's happening, it is happening and you should get on it. Otherwise, you're going to be left behind. But also don't wait too long to get off again.

Emily Lewis: [Laughs]

Lea Alcantara: [Agrees]

Laurie Voss: Like nothing lasts forever. That's what 21 years of web development has taught me. Like when I started, we thought everybody was going to be writing enterprise websites in Java using object-oriented paradigms forever.

Emily Lewis: [Laughs]

Laurie Voss: And we were completely wrong, and then everything seemed to be PHP and we were completely wrong, and everything was going to be – I don't know – everything is going to Rails for a little while, and it was like, "No." Now, it turns out that's wrong too. So right now, React is definitely the trend to get on board, but that train is going to come to a stop sometime and you'll have to remember to get off again.

Emily Lewis: Excellent. Good stuff.

Lea Alcantara: So that's all the time we have for today, but before we finish up, we've got our Rapid Fire Ten Questions so our listeners can get to know you a bit better.

Laurie Voss: Okay.

Lea Alcantara: Are you ready?

Laurie Voss: Yeah.

Lea Alcantara: Okay, first question, introvert or extrovert?

Laurie Voss: Introvert.

Emily Lewis: The power is going to be out for the next week, what food from the fridge do you eat first?

Laurie Voss: [Laughs] I'll drink all of the milk.

Emily Lewis: [Laughs]

Lea Alcantara: What's your favorite website for fun?

Laurie Voss: Hmm, Twitter.

Emily Lewis: What's the last thing you read?

Laurie Voss: Last thing I read was a biography of Alexander Hamilton.

Lea Alcantara: What's the best piece of professional advice you've received?

Laurie Voss: Wow, assume you're the smartest person in the room until somebody proves otherwise.

Lea Alcantara: [Laughs]

Emily Lewis: How about the worst piece of professional advice?

Laurie Voss: [Laughs] Go into enterprise Java, it's the best.

Emily Lewis: [Laughs]

Lea Alcantara: [Laughs] What's your favorite color?

Laurie Voss: Sky blue.

Emily Lewis: If you could take us to one restaurant in your town, where would we go?

Laurie Voss: Mission Chinese Food.

Lea Alcantara: Hmm. What's your favorite board game?

Laurie Voss: Probably Monopoly. It's a boring choice.

Emily Lewis: Not boring, classic.

Lea Alcantara: [Laughs]

Emily Lewis: [Laughs]

Laurie Voss: There you go.

Emily Lewis: Last question, Hulu or Netflix?

Laurie Voss: Netflix.

Lea Alcantara: All right, so that's all the time we have for today. Thanks for joining the show, Laurie!

Laurie Voss: Thank you so much for having me! It's been a blast.

Emily Lewis: In case our listeners want to follow up with you, where can they find you online?

Laurie Voss: By far the best place to reach me is on Twitter where I am @seldo.

[Music starts]

Emily Lewis: Thanks again, Laurie, I really love this conversation. Thanks for joining us.

Laurie Voss: Thanks again.

Lea Alcantara: CTRL+CLICK is produced by Bright Umbrella, a web services agency obsessed with happy clients. Today's podcast would not be possible without the support of this episode's sponsors! Many thanks to Foster Made and Vector Media Group!

Emily Lewis: We'd also like to thank our hosting partner: Arcustech.

Lea Alcantara: And thanks to our listeners for tuning in! If you want to know more about CTRL+CLICK, make sure you follow us on Twitter @ctrlclickcast or visit our website, ctrlclickcast.com. And if you liked this episode, please give us a review on iTunes, Stitcher or both! Links are in our show notes and on our site.

Emily Lewis: Don't forget to tune in to our next episode when we celebrate 100 episodes of CTRL+CLICK CAST. Be sure to check out ctrlclickcast.com/schedule for more upcoming topics.

Lea Alcantara: This is Lea Alcantara …

Emily Lewis: And Emily Lewis …

Lea Alcantara: Signing off for CTRL+CLICK CAST. See you next time!

Emily Lewis: Cheers!

[Music stops]

Timestamp**: **00:51:07