Transcript sponsored by

arcustech

http://ctrlclickcast.com/episodes/code-refactoring

CTRL+CLICK CAST #19 Code Refactoring with Jina Bolton

[Music]

Lea Alcantara: You are listening to CTRL+CLICK CAST. We inspect the web for you! Today we're talking about refactoring web interfaces with Jina Bolton. I'm your host, Lea Alcantara, and I'm joined by my fab co-host:

Emily Lewis: Emily Lewis.

Lea Alcantara: This episode is sponsored by Hover, the place to register your domains with easy, and now, the place to register fun, top-level domains. In fact Hover just launched 56 new domain extensions. Can you believe we used to only have .net, .org, and .com to choose from. Now, we have domain extensions like .tips and .today, and my personal favorite, which came to mind because of today's guest, .coffee. Register for one of these cool domains using the code REFACTOR and you can get 10% off your first purchase on Hover.com.

Emily Lewis: CTRL+CLICK would also like to thank Pixel & Tonic for being our major sponsor.

[Music ends]

Hey Lea, have you seen the news on Devot:ee lately?

Lea Alcantara: Yes, actually. For those that don't know, Devot:ee has relaunched their forums to have more of discussions and stuff like that. Not just the support forums that are for specific add-ons, but like a general discussion forum, and one of the most popular posts right now is titled "Paid Support Options."

Emily Lewis: [Agrees]

Lea Alcantara: Yeah, so right now there's a bit of a discussion going on regarding whether add-on developers should start offering paid support options. What do you think of that?

Emily Lewis: Let's take, for example, one of my favorite add-ons, which is Low Variables for ExpressionEngine.

Lea Alcantara: [Agrees]

Emily Lewis: If in order for me to continue to use it (and my workflows are very dependent upon it) and then I need to pay for support, then I'm going to pay for support because it's kind of essential to the way I'm building my ExpressionEngine CMSes these days.

Lea Alcantara: [Agrees]

Emily Lewis: I also trust Low. He has an established reputation in the industry and in our community. So paying him for support, I trust that he's going to follow through on his end of the deal. So I'm not opposed to it. For me it's a business decision.

Lea Alcantara: Yeah, I would agree. Like as a consumer, of course, I would prefer that I pay just one price. I mean, I'm not saying that there's only one way to do things or whatever, but as a consumer, I'm getting a little bit tired of subscription models.

Emily Lewis: Oh.

Lea Alcantara: It's just like everyone is just another $10 here and another $10 there or whatever.

Emily Lewis: [Agrees]

Lea Alcantara: And that soon becomes like $300 a month, right?

Emily Lewis: [Agrees]

Lea Alcantara: And as someone that really looks at expenses, I guess when you divide it all up over the course of the year, I guess, as a business, you just have to decide whether that's viable for you and profitable for you. But on the other end, too, is looking at it in the developer's perspective: well, you want to make sure that the software is maintained.

Emily Lewis: [Agrees]

Lea Alcantara: And that can't be done if no income is happening, right?

Emily Lewis: Yeah, building an add-on is one thing, maintaining it is a whole other ball of wax.

Lea Alcantara: [Agrees]

Emily Lewis: I mean, I saw the announcement from the guys who made MountEE.

Lea Alcantara: Yeah.

Emily Lewis: And they are discontinuing it partly because of where they've each taken their own careers and they're not really focused on ExpressionEngine these days.

Lea Alcantara: Sure.

Emily Lewis: But that the reality of having to keep up to date with the versions of ExpressionEngine … They said it would be a whole new rewrite.

Lea Alcantara: Yeah.

Emily Lewis: I mean, that's worth time and money.

Lea Alcantara: Oh yeah, absolutely.

Emily Lewis: And I want to see the add-ons that I rely on and feel make my builds better, I want to see them continue.

Lea Alcantara: Yeah.

Emily Lewis: So yeah, I can appreciate what you mean about the subscription model. I feel it every month when they all send me my invoices at the same time for everything, but it's the cost of me doing my business.

Lea Alcantara: Yeah, yeah, yeah.

Emily Lewis: I will say though, and this is not to discourage anyone from investing in add-ons, but if you're using a really good CMS, you can do a whole lot without add-ons too.

Lea Alcantara: Yeah, that's true, that's true.

Emily Lewis: Yeah.

Lea Alcantara: I think part of the issue too, and this is mentioned in the thread too, there's an issue with documentation.

Emily Lewis: Oh yeah.

Lea Alcantara: Because part of the time suck of support situations isn't necessarily because of the bug or even because there was an incompatibility with EE or something, it's just because the person doesn't know what they're doing and they just find it easier and faster to email the developer, or the developer dropped the ball and doesn't have good enough documentation.

Emily Lewis: That happens too often in my opinion.

Lea Alcantara: Yeah, exactly. So for me, I'm definitely pro paying for support as long as there's good documentation in the first place, like if there's enough documentation so that I could troubleshoot it myself, that's really what I want to do.

Emily Lewis: [Agrees]

Lea Alcantara: I don't actually want to ask anybody for help, and secondly, like whether it is fair in terms of like I guess the scale of the problem.

Emily Lewis: Yeah. Well, I mean, and that introduces a whole another part of this argument that if the add-on developers embrace the idea of paid support, what model do they follow, because again using Low Variables, I've never needed support for it. But what if one thing comes up, like an emergency, could I pay for that one-time support need, or is it something that you have to subscribe to?

Lea Alcantara: Yeah.

Emily Lewis: It's a can of worms. It really is.

Lea Alcantara: Oh yeah, and right now, like for those that are interested, you should visit this "Paid Support Options" discussion. We'll link to it in the show notes, and they've been riffing over the different types of ways you could do it. So not all of them were saying subscription models. Some of them are like, "Okay, what if it's a token situation where you pay like a flat fee on Devot:ee of like $100 or $50, and then each question or each level of tier question is like one token, and then until then it depletes it, so it's not an ongoing monthly expense per se." But then there is a situation where if that's the case, are all developers going to follow suit with that flat-fee structure or some people will not, or is there going to be exploration to tokens? Anyways, it's a whole can of worms, and it's way more complicated than we can discuss in like five to ten minutes, but definitely I think people should check out that thread.

Emily Lewis: Yeah, and I think it's important. There are a lot of developers talking about their perspectives and there are a number of consumers.

Lea Alcantara: Sure.

Emily Lewis: Whatever role you fill, get in on the conversation. And a little shoutout to Eric Lamb for pinging us this weekend to sort of take a look at this and it prompted us to discuss it this morning. So thanks Eric.

Lea Alcantara: Thanks Eric.

Emily Lewis: Yeah, so let's get to today's topic. We're talking about refactoring web interfaces with Jina Bolton. Jina is a designer, developer and artist. She currently works at Salesforce as their senior product designer. She's also a renowned speaker and author of *Fancy Form Design* and *The Art & Science of CSS*. Welcome to the show, Jina! Thanks for joining us!

Jina Bolton: Hey, thanks for having me.

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

Jina Bolton: Sure. I live in San Francisco. I've lived here for about seven years now. It's just weird to think about that. [Laughs]

Emily Lewis: [Laughs]

Jina Bolton: Before that I was in Memphis. I lived there for five years, and my hometown is Nashville. I moved out here for a job, and I'm pretty happy in San Francisco.

Emily Lewis: It's hard not to be happy in San Francisco. It's just a great town. [Laughs]

Jina Bolton: Yeah. [Laughs] A little expensive though. [Laughs]

Lea Alcantara: Yeah, yeah.

Jina Bolton: Yeah.

Emily Lewis: Every time I go to visit there, I kind of want to move there, and then I just take a look at how much it costs to rent and find a house and I just don't know how I would do it. [Laughs]

Lea Alcantara: Oh yeah.

Jina Bolton: [Laughs]

Emily Lewis: Unless I had like ten roommates, and I'm a little too old to do that. [Laughs]

Lea Alcantara: Yeah. [Laughs]

Jina Bolton: [Laughs]

Emily Lewis: So Jina, today we're going to talk about refactoring with you. Can you tell me what does refactoring mean when it comes to the web. Is it like a redesign?

Jina Bolton: No, so actually, refactoring is to change the structure of your code without actually changing the behavior, and so it's usually a code cleanup.

Lea Alcantara: [Agrees]

Jina Bolton: But it could be more than that. It's also like organization and trying to move towards a more maintainable code framework just to make like more efficient and easier to work with but doing it in such a way that you're not really drastically changing the web app that you're working on.

Emily Lewis: Is it a situation where refactoring is an ongoing process, or is it something you'd kind of set aside time for focus on, and once it's completed, then you go back to the regular day to day work?

Jina Bolton: So I think most people do it the latter way, but what I advocate for is doing it iteratively and continuously as you're working on the project. Refactoring isn't something that happens very common on static websites or brochureware-type sites, but on web applications that might iterate and evolve the first time. I think it's important to make it a regular part of your process.

Lea Alcantara: So when does refactoring typically come in during a project life cycle? Do you just decide, "Okay, there's this new technique or something, let's just try it out on this one widget"? Or should it start from the very beginning?

Jina Bolton: I think it varies from team to team. The way I prefer to approach it again is to do it continuously, and so anytime I'm making something new and I'm having to put it into place, if there's room to refactor something to make that better, I'll do it then. Or if I'm fixing a bug or changing up a way a component works or looks, then I'll go ahead and refactor it there. And then if I get some extra time (and we like to call a tech debt time, [laughs]) then I'll spend some time doing some refactoring there. But usually, I just try to make it part of anything that I'm doing, whether it's new or old.

Timestamp: 00:09:55

Emily Lewis: So if you're refactoring, and it's making adjustments to the code base without changing the actual functionality or features, I imagine when you're on a team, is the whole team involved, or is one person responsible, and then it's up to you to sort of communicate what's changed and then what needs to be followed moving forward?

Jina Bolton: That's also something that I think depends on your team size and how your team workflow is. In my experience, I tend to be the one doing a lot of the refactoring. I enjoy it. A lot of people don't like refactoring. I actually really enjoy it, but I try to do it in small iterative phases and not try to do it like a complete total overhaul. Because then it's easier for people to review what's been done and we use pull request to help manage that and make sure I'm not just going in there and changing everything around and vice versa. [Laughs]

Emily Lewis: Yeah.

Jina Bolton: So yeah, if you do it in really small iterative phases, then it's a lot more maintainable. It's easy to track and easier to review. So I always try to advocate for doing it in small focused areas rather than doing a big sweep. [Laughs]

Lea Alcantara: Sure.

Emily Lewis: Great. In fact, in your presentation, "Refactoring Web Interfaces," you suggested starting at the element and working outwards. Could you maybe explain that in context to something you maybe done recently?

Jina Bolton: Yeah, so that tends to be like if I'm fixing a bug, maybe it's a UI bug, and I'm inspecting something, and I like to use Chrome Web Inspector. I think it's easy to start at the most small level, so maybe it's a button or maybe it's an icon that's like off or something. I'll start at that, at the small element, and see if it's fixable there. And if it's not, then I go one step up to the parent, and then if it's not fixable there, then I just keep crawling out until I find it, rather than just trying to do something at the top level as that might impact a lot of other things.

Emily Lewis: It's interesting because I try to take that approach, and I ended up finding myself almost going down a rabbit hole. I've ultimately fixed the thing I was trying to fix, but I've already gotten into it and found a couple of other things I could fix, and then the next thing I know, it's like four hours later, and I've been "refactoring" all day.

Jina Bolton: [Laughs]

Lea Alcantara: [Laughs]

Emily Lewis: And happy as a clam because I love doing that kind of stuff, but do you have to put limits on yourself so that you stay focused and stay within budget or within the times for an allotted project?

Jina Bolton: Yeah, I think it's so important to stay focused.

Lea Alcantara: [Agrees]

Jina Bolton: If I notice something else that needs to be fixed, I'll usually like write that down and make a task for it and whatever project management app that your team might be using and get back to it later. Because I think if you're doing code versioning, it's important to keep your commits focused.

Lea Alcantara: [Agrees]

Emily Lewis: [Agrees]

Jina Bolton: Again, it's about being able to review it and it's also easier to revert a small commit than to revert a large commit if whatever you did didn't quite work out properly.

Lea Alcantara: [Agrees]

Jina Bolton: Like if I'm refactoring something, I have to remind myself, "You're just refactoring this one thing, don't touch anything else." Because you will end up in this rabbit hole and realized how long you've spent in there and then you realized you've done a lot more than you should and the next thing you know you have this monolithic branch that takes like forever to review and try to merge. [Laughs]

Emily Lewis: Lea, I think that's a lesson I need to learn. [Laughs]

Jina Bolton: [Laughs]

Lea Alcantara: [Laughs]

Emily Lewis: Lea can speak to the fact that she'll pull something that we've worked on together and it will have probably – I don't know – two dozen things that I've changed. [Laughs]

Lea Alcantara: Yeah, yeah. [Laughs]

Jina Bolton: [Laughs]

Lea Alcantara: Oh, it's all good, it's all good.

Emily Lewis: [Laughs]

Jina Bolton: [Laughs]

Lea Alcantara: But this kind of reminds me of our episode when we had Jonathan Snook on. I mean, he was talking about SMACSS and modular CSS. So we've been talking a little bit about doing things iteratively, little at a time, committing little changes, et cetera, but my question is regarding how do you figure out where you even begin, like what do you tackle first? Is it usually like, let's say, you just got hired and you're working on a particular app, what do you look first to refactor? What should be the priority?

Jina Bolton: I'm a big fan of Sass, and so at first, I determine if the app is using Sass or any CSS pre-processor. If it's not, my first thing is to go ahead and get it started using that because then you can start implementing variables and things like that. If it is already using one, then I can skip that stuff obviously. [Laughs] So then the next thing I would do is I would just start with like things that could be turned into variables.

So these are things like colors or fonts or type sizes, things like that. So maybe I might do a focused refactor on just typography and just look for any repetition or see like exactly how many font sizes I have, and if I can like minimize it down to a really maintainable and modular scale, I'll do that, and then just try to make sure everything is following that scale, and of course, document that into a style guide. If there's not a style guide, then I go ahead and create one and start adding to it over time as I refactor. Sometimes I might start with color if I notice there's a lot of different shades of gray and a lot

of different shades of blue. It's like can I consolidate all that in maintainable color palette and then let Sass do the variations on like darker or lighter versions.

Emily Lewis: So if you pick up something that's not already using a CSS pre-processor and you're going to introduce it to the project, is that something where you leave the original code base and you just add to it, or do you make a lot of modifications to that original code base to then work with the Sass pre-processor and some of the logic that comes with that?

Jina Bolton: Well, if you're using the newer syntax of Sass like they have two different syntaxes. One is white-space sensitive and looks more like Python in that it's like indented to indicate when each selector opens and closes.

Lea Alcantara: [Agrees]

Jina Bolton: And then there's the newer syntax that just looks like CSS, and so if you have a CSS file and just change the extension to .scss and then compile it, it will just compile as a normal CSS file, and so you can kind of get everything moved over into Sass pretty easily just by changing all the extensions and not actually changing any of the code yet.

Emily Lewis: [Agrees]

Jina Bolton: And now you have Sass! [Laughs] It's easy if you're on a Ruby on Rails app or some sort of framework that already has that built in, but if not, then depending on what the type of app is, then you might have to add that in.

Emily Lewis: This is slightly off topic, but I'm curious, do you ever pick up something that was using Less, but you prefer Sass and you want to move it to Sass?

Jina Bolton: Sorry, I haven't had any web apps like that lately. I know when I first used a CSS pre-processor a long time ago when I was at Crush & Lovely, we worked on a project and we used Less,

but I was only using it to do like variations on color because we were doing visualization stuff. It might make it like darker and lighter shades of a color, so I was using Less for that, but I wasn't really using anything else that it came with. The only thing I think recently that was on Less and I wanted to change it to Sass was probably Andy Clarke had at 320 and Up framework that he came up with.

Lea Alcantara: [Agrees]

Emily Lewis: [Agrees]

Jina Bolton: And he did it on Less and I wanted to use it, but I wanted it to be Sass, so I put it over to Sass and then he ended up making that the official Sass implementation, and it's funny because now I understand he's moved over to Sass now. [Laughs]

Emily Lewis: [Laughs]

Lea Alcantara: [Laughs]

Emily Lewis: So did you do that manually or was there some sort of – I don't know – widget or Grunt task that was able to kind of convert it for you?

Jina Bolton: Well, at the time I had never heard of Grunt. [Laughs] So I did it manually, but it was just changing the extensions and then changing the variables from using @ signs, to using $ signs, and then I would compile it and see if it would run, and then it would yell at me about some sort of…

Lea Alcantara: [Laughs]

Emily Lewis: [Laughs]

Jina Bolton: Less does their mixins a little bit differently than Sass does, and their functions are a little bit different. And so I'd have to like kind of go until it would finally compile just properly, and it

took a little bit of time, but it actually got me to learn a lot about Less and I feel more informed in my preference for Sass now from that process. [Laughs]

Lea Alcantara: So that makes me wonder, what is it about Sass do you think that makes it stand out versus others for refactoring?

Jina Bolton: For me it's the community. There are so many tools and so many components and frameworks and things that people build on top of Sass, and just from my perspective of what I've seen, it just seems way more active and way more maintained. And so for me, it's important to use something that I feel is current and I also just think it works better to my workflow. Variables look the way I expect them to look. When I'm searching for variables, it's easier to search by $ sign than @ sign because media queries also use @ signs.

Emily Lewis: Right.

Jina Bolton: Yeah, and I just think it makes more sense for me, but personally, if you're using any CSS pre-processor, I'm happy, but Sass is better. [Laughs]

Emily Lewis: [Laughs]

Lea Alcantara: [Laughs]

Emily Lewis: Well, it introduces the question then, if you aren't using a CSS pre-processor, does that mean that refactoring is more challenging, more difficult? Like can you even refactor effectively if you aren't using something like that?

Timestamp: 00:19:42

Jina Bolton: You absolutely can, but just for me it would make life more difficult if I had to do it without Sass or any CSS pre-processor. I think it just gives you a lot of tools to make your job easier.

I like to kind of use the whole like hammer metaphor where it's like if you have the hammer, it's definitely a lot easier to get that nail in, and you can find other ways of getting it in there.

I also use that metaphor when people try to blame Sass for their output and they talk about, "Oh, my CSS is really messy. Sass did this." No, not really. Sass is just a tool. You could do a lot of damage with a hammer too, but really if you know how to use it, it's really efficient and you can get your job done easier. [Laughs]

Emily Lewis: Do you work in the command line, or do you have a preferred client that you use?

Jina Bolton: So it depends on what it is. Like for most of my websites that I do personally, I might start with Middleman, which is a Ruby-based static site generator, so it's got Sass built in. A lot of the apps I've worked on were Ruby on Rails app, so it's already built in. So I would just write CSS in my text editor, which is Sublime, and then it will compile just because it's built in.

Lea Alcantara: [Agrees]

Jina Bolton: But if I'm working on something that is not built on a framework like that, then I'll often use CodeKit. Like at work, we actually use an Angular-based prototype that I work on, and so we actually use node-sass and it will compile as I save and then it uses like Grunt and it runs all these tests and things. So there's a little bit of back and forth between command line and my text editor.

Emily Lewis: [Agrees]

Jina Bolton: But the only app I tend to use is CodeKit and that's the only one I'm like working on with like something small and static, like maybe I'm doing a single web page or something.

Emily Lewis: Speaking of tools, you've mentioned version control a little bit. What tools are you using, one, if you're doing something personally, but then also what do you use with your Salesforce team?

Jina Bolton: So at Salesforce, we use the enterprise version of GitHub. It's basically just like a way of having GitHub, but having it internally so it's more secure. For personal stuff, if it's just me, it goes into Dropbox, but if it's something that I need to collaborate with other people, like if I'm open sourcing it, then I'll put it on GitHub. So like I worked on the Sass website, and I've put that on GitHub, which has been really awesome because now I don't have to do all the work. People can come in and help me. [Laughs]

Lea Alcantara: [Laughs]

Emily Lewis: [Laughs]

Jina Bolton: Which is really nice.

Lea Alcantara: So there are a lot of things going on when you're refactoring a site, and you mentioned documenting as you go. What do you think is the most efficient way to document? How do you document?

Jina Bolton: Obviously, I'm a big fan of style guides, and sometimes style guides are kind of also something that people put off towards the end.

Lea Alcantara: [Agrees]

Jina Bolton: I think it's important to document as you go so similar with refactoring. I think it's good to add to it as you're creating new things, make sure you modify old things. If you're modifying an old thing that's not in the style guide, then you should put it in the style guide. But also something I've only recently started using is automated generation of things through comments on your CSS, and so there are a lot of tools out there that do this, let's say, StyleDocco is an example. So we did this at Salesforce, so I can write a component, what the markup structure would be and what the description is in the comments in the CSS file and then the page of components in our style guide will

automatically display that component based on what that comment was that I put in with the description. And then we're also exploring Polymer as a prototyping tool, and Polymer basically gives you a way to turn your components into like these little modular pieces that you can kind of move around and like hook into real data, and so all of that would be generated. So we have like one central location where that lives and we don't have to worry about editing the markup in the style guide or editing an app which is edited right there, and then it fabricates. [Laughs] It's pretty cool.

Emily Lewis: That sounds cool. That first one, when you mentioned that you would put like an example of the markup in your CSS comments, did I hear that right?

Jina Bolton: Yes.

Emily Lewis: That's awesome. That's a great idea. I've never actually thought of taking my comments to that level, although it would be really useful for some of our clients, and then the fact that that could then be converted into documentation or style guide, that's fantastic. That was called StyleDocco?

Jina Bolton: That's an example of one, yeah.

Emily Lewis: Okay.

Jina Bolton: There are a few others out there. That's the one I'm seeing or just see talked about the most.

Lea Alcantara: [Agrees]

Jina Bolton: So yeah.

Lea Alcantara: What are those written in? Is it like in Ruby? Do you need to have special tools to run that?

Jina Bolton: So that one is written in node.js, I believe.

Lea Alcantara: Okay.

Jina Bolton: I think KSS might be done in Ruby because it's a gem that you install. I might be wrong on that, and I'm trying to think of the other one I've seen. There's one I saw that was just sort of a drop in like JavaScript file, and you would have a page set up, but I can't remember what it's called, I'm sorry. Yeah. [Laughs]

Emily Lewis: Now, in terms of the style guide that you mentioned that you're sort of documenting as you go, is it ever like a final style guide and then it becomes versioned itself, or is it just kind of a living document that you're just always updating as you're refactoring?

Jina Bolton: So I love using the term "living style guide." I think a lot of people referred to their style guides as living just because it's up on the web, but they don't maintain it as they update the app.

Lea Alcantara: [Agrees]

Jina Bolton: And so in that case, I don't really feel like it's truly living. But yeah, I try to make my style guide as living as possible, and what I mean by that is if I change or add a new color, I shouldn't have to go than open up my style guide and add that color, like it should just be there. And so I did this using YAML data on the Sass website, but at Salesforce we're exploring doing that with JSON and so we just have like one central location where you can like put in your colors, your fonts and things, and then you don't have to worry about adding the new swatch or that new value into the style guide. It would just display automatically.

Emily Lewis: And let's say you've made an update, you've refactored something, you've adjusted the code base, the style guide is now updated. What's the next part of communication? Do you do

like a mass notification to all your users of the app, or is it up to them to sort of keep tabs on what's been updated in the style guides? So they know what might have been changed?

Jina Bolton: I think it depends on what the app is.

Lea Alcantara: [Agrees]

Jina Bolton: So like for something small like the Sass website, it just all on GitHub, so I don't really do a communication when it's updated, and I don't think it necessarily needs to be communicated for something on that small scale. So what we're doing at Salesforce, what we do could impact a lot of different teams. I think we have an Android team, an iOS team, a web framework team. We have all the prototyping teams, and so we're actually working on improving that workflow now and what we've currently resorted to is central repository that has each new variables there and then we have a tool that we built that generates all the different files needed for each of these things. So it might be a Sass file, it might be an XML file for the Android app, it might be plist file for the iPhone app, but it's all generated from that one central location.

But this is in a repository, the different teams know it's there to pull in when they're ready for it, but that they can do it when they're ready for it so we don't just like to start and breaking things by like changing it automatically for them.

Emily Lewis: Right.

Lea Alcantara: [Agrees]

Jina Bolton: So at that point, we would probably do like a communication like, "Hey, a new version is out." Then when those teams are ready they can pull it in to their code.

Emily Lewis: And how often are you totally working on your own, and how often do you have to bring in someone from, let's say with Salesforce, the Android team, because perhaps something you

do would affect them? Is there a collaboration involved in refactoring? Or is it really you come up with a solution and then it's up to those teams to let you know or ask their own questions after?

Jina Bolton: We are only just recently starting to communicate with other teams. I'm so fairly new to the team, but like so far, the experience seems to be that we would work with the teams throughout the process, but not like side by side, like if we wanted to change things, we would have to just let them know that things are going to change. We also have this internal tool that we built that hosts like all our designs, and so we've recently communicated to all the different teams that any time you need to see like the newest spec or the newest design for something, you would go here and see that latest and greatest. So personally, I'm making it my own personal mission to try to communicate with other teams more, but there's a lot of different people.

Emily Lewis: [Agrees]

Jina Bolton: And so I'm still learning who all these people are that I need to talk, but yeah, it's been a good learning experience for me in understanding how something I do could affect something somewhere else.

Emily Lewis: [Agrees]

Jina Bolton: And actually, the team that I'm on right now is called Systems, and like our whole goal is to try to avoid issues like that. Like we try to design in a very systemic way and make sure that everything is aligned, but any changes that you make, we need to consider like how those changes might affect something somewhere that might not be totally visible at the beginning, but like maybe there's some page somewhere or some UI somewhere that we didn't think about that it could affect.

Lea Alcantara: So why don't we get into a little bit more of the nitty gritty of the actual refactoring. We got a question from our listener, Kevin Lozandier, and he asked, "What's a particular code smell that drives you nuts because of its frequency that's easily avoidable?"

Timestamp: 00:30:08

Jina Bolton: [Laughs]

Emily Lewis: That's the first time I've ever heard of "code smell." [Laughs]

Lea Alcantara: Yeah, me too. [Laughs]

Jina Bolton: Oh yeah. It's pretty common on the engineering side. It's less common a phrase used on the design side, but yeah [laughs]. So a big one is just letting your colors get out of hand. I see it so many times where people might like not want to document a new color, so they're just going to add it in and then you've got like 30 different shades of dark blue. [Laughs]

Lea Alcantara: Yeah, yeah.

Jina Bolton: So that's a pretty common one. I think one thing that drives me crazy is just overly-specific selectors.

Emily Lewis: [Agrees]

Jina Bolton: Because I try to make my selectors as small and flat as possible, and I think a lot of people that first get into Sass kind of see, "Oh, you can nest with Sass," and so they make it match their HTML structure, but in the end you end up with the sloppiest, like longest CSS ever. You have to try to keep things as small as possible, as flat as possible, and that also makes it more usable, and so you don't have to duplicate anything or do as many overrides of things because you can just kind

of reuse these little components throughout. So that's something: overly specific or styling based on where it lives rather than what it is drives me crazy. [Laughs]

Lea Alcantara: [Agrees]

Emily Lewis: So Jina, when you were sort of describing some of those little nuances, it sounded like you're aiming for like a DRY principle to try not to repeat yourself too much.

Jina Bolton: Yes.

Emily Lewis: Are there any DRY principles that you really follow when you're building a UI interface that might be unintuitive for those who are unfamiliar with something like Sass?

Jina Bolton: Oh, that's a good question. So I think like I definitely try to follow the DRY principle of not repeating yourself, and at work we always use the phrase, "single source of truth," and that kind of refers to making sure that things only live in one place.

Emily Lewis: [Agrees]

Jina Bolton: However, I do think sometimes people over abstract things. And I forgot who, but it might have been Chris Eppstein that once said this to me, but it was to build it and then if you find yourself really using it, maybe making a note of it, or see like if it's definitely… and I'm probably saying this totally wrong, but I'm trying to remember. It was something like it wasn't until the third time that you see something repeated that you would abstract it.

Emily Lewis: Oh, okay.

Jina Bolton: But I don't remember, I don't quite remember what that second middle ground was. [Laughs] So yeah.

Emily Lewis: Well, I find myself in that situation sometimes too where I'm going into it and I'm trying so hard to be DRY that I'm probably creating more work for myself with things that I wouldn't actually reuse somewhere. [Laughs]

Jina Bolton: Right.

Emily Lewis: That they're not reusable.

Jina Bolton: Yeah.

Emily Lewis: That they aren't things that are modular that I would drop in in every project or something like that.

Jina Bolton: Yeah, and so there's this tendency to want to abstract any and all instance of repetitive elements, but sometimes if you'd overdo that, you make things harder to find or harder to look for it. Like you have to look somewhere and then you see, "Oh, actually, it's over here," and then you see, "Oh okay, actually this piece is over here because it's like abstracted so many times.

Emily Lewis: [Agrees]

Jina Bolton: So I try to like find the balance and abstracting things where they make sense, but not abstracting things just for the sake of abstracting things. [Laughs]

Lea Alcantara: So as we wrap up this discussion, just a few final questions. What do you think is the biggest roadblock for someone to attempt refactoring an interface?

Jina Bolton: I think a lot of people try it … When they think about refactoring something, they think about having to clean up as much of it as they can all in one go, and in fact, it ends up getting really overwhelming.

Emily Lewis: [Agrees]

Jina Bolton: And I can tell you from my personal experiences, when you do that, you're going to end up with this branch that sits around and never gets merged because it's just become way too big of an undertaking. It's sort of like I kind of use a lot of metaphors, sorry about it, like around houses and like tools, but like if you are wanting to remodel your house.

Lea Alcantara: [Agrees]

Jina Bolton: You wouldn't try to remodel every single room all at once. You would just remodel like a room at a time, and that's how I would approach it, like you just focus on one room, with the kitchen, and then you're done with the kitchen, now you do the bathroom. You wouldn't try to like refactor everything. So I think with web interfaces, it's common to want to fix all the things like all at once.

Lea Alcantara: [Agrees]

Jina Bolton: And I think that would make you want to hesitate like how do I start. If you already know going in, "Okay, I'm just going to do color. I can't touch anything else but color. I'm only doing color," then I think it's easier to have a sense of where to start, and you will have an easier time actually finishing the task.

Emily Lewis: And the way you described that, I'm thinking of it in terms of myself. I think I've erred or not erred, I've made a rather unfortunate decision to do too much at once, and it does create, like you said, a branch that's just too massive or there are so many changes within a single commit that it's harder to review. But also just me setting aside the time in my week. If I'm approaching it thinking, "I'm going to clean up everything," then I'm like, "Oh, I've got to wait until I have a chunk of four hours as opposed to, "Oh, if I just handle color, I can do that in an hour or hour and a half, then no problem, I can put that into any schedule."

Lea Alcantara: Yeah.

Emily Lewis: Since you have such a knowledge of Sass and you're so tied to the community, what are your top learning resources you recommend for someone who's just getting started, not just for Sass but the whole concept of a CSS pre-processor.

Jina Bolton: Things that I would recommend, obviously, there is all the documentation on the Sass website, but I totally recognize that that documentation isn't amazing. It needs work. If you want to help me with that, please help me. [Laughs]

Lea Alcantara: [Laughs]

Emily Lewis: [Laughs]

Jina Bolton: But also, I like to read … his last name escapes me and I apologize, but his first name is Hugo [Giraudel]. He's on Sitepoint, and he always writes these great articles about Sass and features and like architectural tips and like how to approach certain common problems that you might face. So that's been like really useful for me to read, but that might be a little bit more on the intermediate to advance level for like beginners.

I would say Chris Coyier does a lot of talk on CSS pre-processors on *CSS Tricks*. That's a pretty good place to start. I watched this video on, I want to say it was on Code School that I thought, "Oh, I don't really need to watch this because I already know Sass." But I got things out of it that I didn't even know you could with Sass, and it was still like at a very human-friendly level of just teaching it without sounding like way too technical. I also think there's this tool called SassMeister that has been helpful for me to use when I'm teaching Sass internally at work, because SassMeister lets you play with Sass without having to get it set up, work within a web application.

Emily Lewis: Oh.

Jina Bolton: So it's pretty cool because then I can teach like certain principles of Sass in like how to use it, and then the people I'm teaching can start using it and see what it's going to output side by side, and then they can save it on GitHub or if they wanted to bring it into a project later, they could. But it's really cool for just playing with it without actually having to worry about like the setup. What else? Yeah, there is a lot out there right now. [Laughs]

Emily Lewis: What about for getting into refactoring and kind of building a good workflow for refactoring, are there any resources out there that you turn to?

Jina Bolton: I don't know that there are a whole lot for like web interfaces like in terms of HTML and CSS. There are a lot of articles that I've read, and I read SMACSS, and I check out a lot of what people are doing with their different framework and things such as Foundation and Bootstrap. I don't use any of these tools, but I'd like to kind of see like any time they make a change to this framework, what kind of changes are they making and why did they make those changes.

But there's not a whole lot of actual tools that I can think of other than just like all the different things people are contributing towards it. In the Sass community, there's a lot of things people are making to maintain your colors easier, things to generating the style guide easier or things to make our layout system easier. Like people are creating all these different tools to help make that process easier to maintain, and there are tools out there that do that without Sass too, but obviously, I kind of look towards the Sass one. [Laughs]

Emily Lewis: [Agrees]

Jina Bolton: Yeah.

Lea Alcantara: All right, so before we finish up, we've got our rapid-fire ten questions so our listeners can get to know you a bit better. Are you ready?

Jina Bolton: Okay. [Laughs]

Lea Alcantara: All right.

Jina Bolton: Okay.

Lea Alcantara: The first question, Mac OS or Windows?

Jina Bolton: Mac OS.

Emily Lewis: What is your favorite mobile app?

Jina Bolton: Oh, Instagram.

Lea Alcantara: What is your least favorite thing about social media?

Jina Bolton: Hashtags. [Laughs]

Lea Alcantara: [Laughs]

Emily Lewis: [Laughs]

Jina Bolton: Even though I use them. [Laughs]

Emily Lewis: What profession other yours would you like to attempt?

Jina Bolton: Interior design.

Lea Alcantara: Cool. What profession would you not like to do?

Jina Bolton: One that I've already had to do before, anything sales related like telemarketing. [Laughs]

Emily Lewis: Who is the web professional you admire the most?

Jina Bolton: Dan Cederholm.

Lea Alcantara: What music do you like to code to?

Jina Bolton: Witch House. It's a weird genre of music that's a blend of lots of others, but yeah. [Laughs]

Emily Lewis: What's your secret talent?

Jina Bolton: Oh, secret talent? I have this weird knack for like I'll talk about something, and then I'll check Timehop and then see that whatever I'm talking about happened exactly one year ago today.

Lea Alcantara: [Laughs]

Emily Lewis: [Laughs]

Jina Bolton: It happens a lot. It's really weird. [Laughs]

Lea Alcantara: What's the most recent book you've read?

Jina Bolton: *Sass For Web Designers* by Dan Cederholm.

Emily Lewis: Lastly, *Star Wars* or *Star Trek*?

Jina Bolton: *Star Wars*. I like both, so that's hard. [Laughs]

Lea Alcantara: [Laughs]

Emily Lewis: [Laughs]

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

Jina Bolton: Thank you.

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

Jina Bolton: So I'm @jina at Twitter and Instagram and GitHub [laughs] and some other sites. Yeah, my website which isn't really much right now is jina.me or also sushiandrobots.com, they go to the same place. I might relaunch that site soon. But yeah, mostly Twitter and Instagram @jina.

Emily Lewis: Great. Thanks again, Jina! It was so great talking to you.

Jina Bolton: Thank you.

Lea Alcantara: [Music starts] We'd now like to thank our sponsors for this podcast, Hover and Pixel & Tonic.

Emily Lewis: We also want to thank our partners, Arcustech, Devot:ee and EE Insider.

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.

Emily Lewis: Don't forget to tune in to our next episode when Jason Varga is joining us to talk about ecommerce for Statamic. Be sure to check out our schedule on our site, 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: 00:41:34
