http://ctrlclickcast.com/episodes/progressive-enhancement-revisited

---

CTRL+CLICK CAST #067 – Progressive Enhancement, Revisited, with Aaron Gustafson

[Music]

Lea Alcantara: From Bright Umbrella, this is CTRL+CLICK CAST! We inspect the web for you! Today our friend, Aaron Gustafson returns to the show to update us on progressive enhancement. I'm your host, Lea Alcantara, and I'm joined by my fab co-host:

Emily Lewis: Emily Lewis!

Lea Alcantara: This episode is brought to you by Craft Commerce, a brand new e-commerce platform for Craft CMS. If you're a web shop that likes to create custom-tailored websites for your clients, you're going to love Craft Commerce. It's extremely flexible, leaving all the product modeling and front-end development up to you, and it's got a simple and intuitive back end for content managers. To learn more and download a free trial, head over to craftcommerce.com

[Music ends]

Emily Lewis: Today we are revisiting a topic that is as important today as it was three years ago when Aaron was first on the show, progressive enhancement. As author of *Adaptive Web Design* and former manager of the Web Standards Project, Aaron is passionate about web standards and accessibility. He has been working on the web for nearly two decades and is a web standards advocate at Microsoft working closely with their browser team. Welcome back to the show, Aaron.

Aaron Gustafson: Hey, thanks for having me.

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

Aaron Gustafson: Oh gosh, let's see. Well, probably the most recent news is my wife and I just finalized the adoption of our first son, Oscar.

Emily Lewis: Yay!

Lea Alcantara: Yay!

Aaron Gustafson: So anyone who follows us on Facebook is having to deal with the deluge of cute baby photos.

Emily Lewis: They're super cute though.

Lea Alcantara: [Laughs]

Aaron Gustafson: So that's the most recent update.

Emily Lewis: [Laughs]

Lea Alcantara: [Agrees]

Aaron Gustafson: He's adorable and we take no credit for him.

Emily Lewis: [Laughs]

Lea Alcantara: [Laughs]

Aaron Gustafson: So we just lucked out. We won the baby lottery. No, he's an awesome kid.

Lea Alcantara: Nice.

Emily Lewis: So I mentioned in your intro or your bio that you're author of *Adaptive Web Design*, it actually came out in its second edition this past December. I'm curious, what's the process with like updating a book?

Aaron Gustafson: That was kind of an interesting thing. I knew I wanted to update the book, and I've kind of wanted to update the book for like the last two years probably, but I hadn't been able to kind of carve out the time yet with client work and all of that other stuff, and then things sort of shifted when I took the position in Microsoft and it actually cleared up a little bit more of my free time to be able to do some writing, and so my initial intent was to come into the book and just update a little bit here and there, like I hadn't really tackled responsive design in the context of progressive enhancement.

I had only talked really about media queries. I hadn't talked really at all about single page apps and that sort of stuff, so I had an idea that these were some of the things that I wanted to tweak, and so my guy going into it was, "Okay, I'm going to take this, roughly 135-page book, and reformat some stuff and add some new stuff in and maybe we'll end up at around 160 pages or so." So that was my initial intent, but once I got in there, I realized that there were a lot that had changed in the way that I approach progressive enhancement since I wrote that first book and 2010 is when I originally wrote it.

Emily Lewis: [Agrees]

Lea Alcantara: Right.

Aaron Gustafson: So it had been five years at that point since I had originally written a book, and so I kind of took it back to the drawing board and said, "Okay, how would I want to write this now? How do I want to frame things differently? How have I seen the arguments for progressive enhancement, work within large organizations that I've had benefit of consulting with in kind of the intervening years, and how have things changed on the web, and that sort of stuff, and so I basically scrapped the original book.

Lea Alcantara: [Agrees]

Aaron Gustafson: Even though all of the lessons in it were still completely valid, I just wanted to approach things differently. Whereas the first book had a full chapter that was just devoted to accessibility, I realized that that really does accessibility a disservice and I wanted to integrate accessibility throughout the entire book and really not make it feel like an add-on.

Emily Lewis: [Agrees]

Lea Alcantara: [Agrees]

Aaron Gustafson: It needed to be part and parcel of kind of the process, and I also wanted or I felt like I didn't do justice to where progressive enhancement and content strategy and copywriting meet as well, and I kind of glossed over that in the first chapter of the first edition, like this is something you need to think about, but as I've come to better understand progressive enhancement and better understand experience design period.

I know that experience design begins with the content that we write and every experience begins with that conversation that we're having with our users, so I wanted to devote an entire chapter just to kind of content strategy and copywriting and how that fits into the progressive enhancement continuum as well. So there were a lot that I ended up restructuring.

I have way less detailed and specific code examples. Whereas I followed the development of a specific site in the book, in this one I actually highlight great examples of progressive enhancement from the wild, which I think is a little bit more beneficial rather than kind of talking about a single project that kind of shows, "Here's how a bunch of different people are approaching it."

Emily Lewis: [Agrees]

Lea Alcantara: [Agrees]

Aaron Gustafson: So I still wanted to keep it very much a philosophy book as opposed to a technique book in which I feel like I was successful with and that was one of the things that I felt was great about the first edition and why it had such longevity. I mean, people were still buying it up until the point that the second edition went on sale, so I wanted to keep that, but yeah, I ended up having to rewrite the whole thing, and what was supposed to be a 160 pages ended up being like 260-some odd pages. [Laughs]

Emily Lewis: [Laughs]

Lea Alcantara: Wow!

Aaron Gustafson: But I mean, it was a lot of fun rewriting it. It was kind of nice to get to revisit that. I know Jeremy Keith likes to say that he doesn't like to update his books because you can't step in the same river twice, but I feel like there is something really nice about getting to kind of revisit it and where as it is the second edition, philosophically, it pulls from that first edition, but there's only probably 20% of that first edition that's still in the second edition. Everything else has kind of been rethought and revamped and it could have been its own book.

Lea Alcantara: Interesting.

Emily Lewis: Yeah. I feel like you did achieve that goal of keeping it. Philosophical sounds almost like it might be hard to read. It's not very hard to read, but I think progressive enhancement, and even just web design and development in itself, I find so much more a mindset an approach and a process today having 20 years under my belt than I did in the beginning when I was just focused on the technical aspects of learning how to code.

Lea Alcantara: Right.

Emily Lewis: And so I appreciated that sort of approach to these concepts because I feel like that's the harder stuff, to really grasp the concepts and the whys versus building the skills to execute.

Lea Alcantara: I do think this is kind of like a reflection of our current times as a designer or developer in like the culture of the web these days. Five or six years ago when the first edition came out, I think people were still kind of figuring out how the web was going to be, and then now that there are kinds of standards that have been set, it's kind of like, "Okay, now, we've got these skill sets, how do we apply that holistically? How does this actually apply from the beginning to the very end of the project instead of like some single-focused, maybe only front-end focused part.

Emily Lewis: Right, really discreet.

Lea Alcantara: Yeah, yeah, exactly. When I looked at the second edition that Aaron wrote, it was just really stark even before I dove into the book that it was different because the table of contents was completely, completely different. It did feel like a different book.

Emily Lewis: So Aaron, I think you'd be useful. I know we did address this when we first had you on the show, but for any of our new listeners, I think they might be curious why you named your book *Adaptive Web Design*, but that in industry term, it's more commonly known as progressive enhancement. Is there a reason why you named your book that way, a difference between the two concepts or phrases?

Aaron Gustafson: So originally the reason that I chose to not have progressive enhancement as the title is I wanted to disambiguate the book from Filament Group's *Designing for Progressive Enhancement*, which had only came out a year or two earlier, and so I love and respect that group. I

love the folks on that team. I think they're awesome. I think they do great work and I didn't want to cause confusion if they were on the same bookshelf in a Barnes & Noble or what have you. It was partially out of respect for that book that I didn't want to have progressive enhancement as the title. The other sort of facet of that is that progressive enhancement sounds so clinical.

Emily Lewis: [Agrees]

Lea Alcantara: Right.

Aaron Gustafson: I mean, it's absolutely dead on for what it is that we're doing, but it does sound so clinical, and so I wracked my brain and actually worked with my original editor, Krista Stevens, back when the first edition came out to once the book was written to come up with an appropriate title for it, and we came up with a bunch of different things, and that I think we eventually settled on Adaptive Web Design: Crafting Rich Experiences with Progressive Enhancement because it had progressive enhancement in the subtitle, but it kind of spoke to a larger idea of a web that could anywhere, that could take any shape.

Emily Lewis: [Agrees]

Lea Alcantara: Right.

Aaron Gustafson: And interestingly enough, it was probably somewhere in the back of my mind, but it wasn't at the forefront of my thinking when I initially went in there to come up with a title but actually John Allsopp way back in *A Dao of Web Design* referred to the approach as adaptive web design. So it actually has a nice sort of synergy with his very seminal work, which gave rise to responsive design and so on and so forth.

Emily Lewis: [Agrees]

Aaron Gustafson: But one thing I will say is that they definitely became because Ethan's (Marcotte) book came out at about the same time as the first edition. There was some confusion as to what was adaptive web design versus responsive web design and people felt like they were at odds, and I think it was because I didn't really address responsive web design because I hadn't really thought that deeply about it yet and kind of figured out how it would fit into the progressive enhancement continuum. People thought that they were at odds, which they weren't and they can be very much part and parcel of the same approach, especially if you're taking responsive web design approach from a mobile-first perspective.

***Timestamp**: 00:10:10*

Emily Lewis: [Agrees]

Aaron Gustafson: So that's one of the reasons that I definitely wanted to include that in the second edition of the book to ensure that everybody was aware of that.

Emily Lewis: And let's just put the definition out there, what is progressive enhancement in terms of what does that mean to a website user?

Aaron Gustafson: So progressive enhancement is all about the approach that you use to create an experience, and it's about realizing that experience is a continuum. Not everybody is going to have the exact same experience of your website, of a particular interface within your website, or what have you, and to build that experience, layer upon layer, to create something that can something that can work for anybody, and being okay with the fact that that experience may vary from person to person or even context to context with the same person, whether they're on a laptop or a desktop may have a slightly different experience when they're on their phone, and that's okay. So that's basically what

it's about, and it's about not putting particular technological restrictions on the use of your site, so you're not creating artificial barriers to access.

Emily Lewis: Right. So you mentioned the device scenario, but it goes in the other direction as well with someone who might be using a text-based browser or someone who's using assistive technology or someone who utilizes the web differently like they prefer navigating with the keyboard versus the mouse kind of thing. So it's creating that fundamental experience for everybody regardless of what they're using and then enhancing that as, I guess, their device or usage shifts

Aaron Gustafson: Yeah, as the device becomes capable of doing a particular thing. I mean, I think kind of a classic example of baked-in progressive enhancement is some of the HTML5 input types, right?

Emily Lewis: [Agrees]

Aaron Gustafson: You've got input type equals email. If you use that and somebody happens to be browsing your site in links, which Lynx, which is an ancient browser at this point, it has no idea what to do with that. Similarly, a lot of desktop browsers are just going to present a standard text field, and that's what the fallback is because any unknown input type defaults to being text, and anybody with fat finger types like I do and accidentally writes "rdio" instead of radio for an input type and wonder why all of a sudden their radio button turned into a text box has experienced that.

So basically any input type that's not recognized becomes a text field by default. That's fault tolerance in HTML. But that creates an opportunity for browsers that understand that email input type, they can apply validation against the email field. They can perhaps in a virtual keyboard context like a tablet or a phone, they can present a specific keyboard that is optimized for the user to enter

that information, and so it's creating an enhancement for the users who can take advantage of it, but it's not the only experience.

Emily Lewis: [Agrees]

Aaron Gustafson: It's allowing people to still be able to enter their input or enter their email address into that input field, even if they're on IE6.

Emily Lewis: [Agrees]

Lea Alcantara: Okay. So the scenarios that you just explained sounds to me a little bit like the term a lot of people also use, which is graceful degradation, however, it might not necessarily be the same thing as progressive enhancement. So is there a difference between progressive enhancement and graceful degradation?

Aaron Gustafson: They're very related approaches in that all progressively-enhanced designs are going to degrade gracefully.

Emily Lewis: [Agrees]

Aaron Gustafson: So all progressive enhancement is graceful degradation.

Lea Alcantara: Okay.

Aaron Gustafson: But the inverse is not true, not all gracefully-degrading sites and interfaces are progressively enhanced, so for instance, you could have a site that checks to see if you're in a particular browser and if you're not in a browser that they support or if you're in a browser that they don't' recognize, they could put up a page that blocks you from being able to access the thing and provides you a nice little message that says, "You need to use this browser in order to access our

site," and you haven't technically throw an error, you've degraded gracefully, but it's not progressive enhancement.

Emily Lewis: Right. [Laughs]

Lea Alcantara: Right, okay.

Aaron Gustafson: So that's a roadblock, but it's not throwing an error in the browser, if you follow.

Lea Alcantara: Okay.

Aaron Gustafson: So that's kind of a stark example, but it's a pretty good example of graceful degradation and an approach that you could take that would be legitimate graceful degradation, but it's not progressive enhancement.

Emily Lewis: Tell me, is this the same? The other day I had JavaScript disabled in my browser and I went to Facebook, and you can't use Facebook without it. It gives you a message. Is that the same thing basically?

Aaron Gustafson: Yeah, absolutely. "You must be this high to ride." You have to have this technology. Our stack is requiring this in order to be able to access the site, and yeah, there was actually a great post (The Web Isn't Uniform) on Medium just the other day from Karolina (Szczur). I'm going to butcher her last name. She's @fox on Twitter, and basically, she was bemoaning this wall that we have essentially created by relying 100% on JavaScript on the front end and how many sites just fail to work.

Emily Lewis: [Agrees]

Aaron Gustafson: And there are developers out there who are okay with that, and I have to agree with her. I mean, I think from a developer perspective because I do consider myself a developer, that's not building a robust site because JavaScript is not a guarantee.

Emily Lewis: [Agrees]

Aaron Gustafson: Like I love JavaScript, I write it all the time, but thinking that it's going to run everywhere and every scenario is just delusional.

Emily Lewis: [Agrees]

Aaron Gustafson: I mean, we don't control the execution environment of our code, and even if you lock, if you're building an internal thing and it's on your intranet or whatever, you still don't control the browsers that are accessing it, or even if you have a company-standard browser and you're optimizing it for that, if somebody is out on the road and they're tethered to their phone or something like that or they're on hotel Wi-Fi, like all of these scenarios within which like your JavaScript framework either may not download or may take forever and execute.

Emily Lewis: [Agrees]

Aaron Gustafson: And so we can't rely on JavaScript always being available, and I certainly don't think we should be using JavaScript to supplant the things that browsers do, which is download web pages and render content, and yet we have all these frameworks that seem hell bent on "I'm taking over for the browser," and I frankly don't get it. It seems like you're just throwing additional code at something, and I don't know, I feel like a lot of times we as a development community are way more focused on enjoying our jobs and challenging ourselves and learning new and shiny things than we

are in solving our user's problems, and so we make things easier for ourselves and make our users pay for it.

Emily Lewis: [Agrees]

Lea Alcantara: [Agrees]

Aaron Gustafson: And I think that's a bad way to go. You know what, we should be doing everything that we can to make our users' lives easier, and I don't feel like we are.

Emily Lewis: Yeah, I agree with that. It's our users, who are using the website, but also the clients who are hiring us to build sites that they can count on, and we've got a Frankensite that we inherited however many years ago, and it relied on Bootstrap.

That's who the developer used Bootstrap, and we are finding all these little things over the course of the years that are failing now that in order to uncouple, we have to undo the whole thing and basically redesign the site, and it's incredibly frustrating to communicate this to this client who's made an investment in the site and is committed to making an ongoing investment, but they feel they just had this site redone.

And now they're having problems with things in Bootstrap failing because it's no longer supported, and I frankly can't stand Bootstrap so I pretty much write from the scratch when they let me, and the users of the site are suffering and the client is suffering. And we as developers are suffering because the original site was built dependent on a framework that was dependent on a bunch of other things that half aren't even supported it and haven't been updated in year. It's crazy.

Aaron Gustafson: Yeah, and that's for a majorly "supported" project, right?

Emily Lewis: [Agrees]

Aaron Gustafson: Like there is huge backing behind that, and there are literally millions of sites that are on Bootstrap, and that's rather pathetic.

Emily Lewis: [Agrees]

Lea Alcantara: Right.

Aaron Gustafson: Not that they're made it on Bootstrap, but that it causes that many problems because it's not being maintained and it's not easily upgradeable or what have you.

Emily Lewis: So Lea had found an article from 2013. Tom Dale is the guy who did Ember.js.

Aaron Gustafson: Yeah.

Emily Lewis: And he basically railed against progressive enhancement.

Lea Alcantara: Yeah. [Laughs]

Emily Lewis: And even reading it, trying to understand his perspective, I just quite frankly don't. You're in a unique role being an accessibility advocate from Microsoft, you must have conversations with people who have this different perspective, do you at all understand where they're coming from? Is there anything that they have some sort of valid point somewhere in that approach?

Aaron Gustafson: I can understand where some people are coming from in that, and so that particular post was Progressive Enhancement: Zed's Dead, Baby, which was like he's basically bemoaning the death of progressive enhancement because we're in this new world where effectively we've got a virtual machine in a browser and we can do JavaScript or use JavaScript to do everything awesome. Now, what's interesting about that is two years later, he came out with another post that

was basically all about Ember FastBoot (Inside FastBoot: The Road to Server-Side Rendering), which is doing server site rendering in Ember. [Laughs]

Emily Lewis: [Agrees]

***Timestamp**: 00:19:59*

Aaron Gustafson: And so he basically did a complete 180 and without saying, "No, you should be doing progressive enhancement," he's saying, "No, you should be doing progressive enhancement." You should effectively be doing what's come to be known as isomorphic JavaScript where it's rendering the Ember stuff on the server as well so that you always have a response to a request as opposed to abject failure if JavaScript is not available.

But back to your question about that, like is there validity with his perspective... I can see where folks are coming from, and I have to imagine that in a lot of cases, this is when there are folks who are coming to the web from kind of a traditional CS back-end programming — maybe Java, Python, that sort of stuff — where they're building applications and it's really important that when you're working in a medium that you understand that medium.

And I think that's the problem in a lot of cases, folks are coming over and they don't understand how the web works and therefore, they're not able to accommodate this. Some people refer to them as quirks or idiosyncrasies of the web. I can consider it's sort of the beauty of the messiness of the web and its ability to kind of pore through all of these different cracks to reach users where they are and in the ways that they need to be reached.

But I think that's sort of the fundamental problem, and we used to see this. If you look back to the early days of the web when some developers were starting to build websites, and maybe they were

coming from Java and they were generating web-based interfaces using JSPs or whatever. A lot of times they didn't put much thought into the HTML that they were writing or the CSS that they were writing or the JavaScript they were writing. They just knew that it worked and they didn't care about the semantics and they didn't care to learn the languages because they were seen as less than or toy languages. I often heard that like toy language referring to JavaScript.

Now, JavaScript is a big freaking deal and some people are really into JavaScript, but they're still not understanding the sort of the beauty and power of HTML or of CSS. If you listen to the industry, it kind of had taken Stephen Hay's lead on this and refer to a lot of these folks as Stack Overflow developers.

Emily Lewis: Right.

Aaron Gustafson: I mean, sorry, Full StackOverflow Developers.

Emily Lewis: [Laughs]

Lea Alcantara: Yeah. [Laughs]

Aaron Gustafson: And that's kind of the way it's gone where, "Oh, this works. I'm moving on, like I fixed this problem, but I don't know why it works. I just know that it works."

Emily Lewis: [Agrees]

Aaron Gustafson: Which, I mean, we ran into that same stuff when Dreamweaver was the way everybody built sites too, where they had no idea what they were doing, creating all of these divs and moving them around in the WYSIWYG editor and all this stuff was absolutely positioned and yadda, yadda, yadda. It's a cycle.

It happens over and over and over again on the web, but I think that more people start to understand the web and understand the tools that we have to build web pages, the better their work becomes and the more capable their work becomes of being able to go anywhere.

I mean, if you're thinking about the semantics with a page, if you're thinking about the conversations that you're creating and that you're enabling, you increase the accessibility of your page, and you know what, you're going to make your site work better in the not too distant future of headless UIs where more and more people are interacting with the web via tools like Siri and Cortana and Alexa and so on and so forth where there will be no visual design. All they'll be doing is interacting with the site via voice and text.

Lea Alcantara: You know what, what it sounds like to me as you are explaining some of people's perceived drawbacks and their main thing is like, "Oh, while it works, it's fine." It feels like the core problem is that some developers mistakenly think that they're the end user.

Emily Lewis: Right.

Lea Alcantara: Like that they're the only audience, and it's kind of like a bad perception type of thing where just because you're building it doesn't mean you're building this for yourself.

Aaron Gustafson: Absolutely.

Lea Alcantara: Right?

Aaron Gustafson: Yeah, it's a huge problem within the web industry, and I would say within the app industry as well where we're building stuff for ourselves.

Lea Alcantara: Right.

Aaron Gustafson: I mean, do we really need another photo-sharing app? Do we really need another hook-up app?

Emily Lewis: Right. [Laughs]

Aaron Gustafson: I mean, you see some of these startups that come out that are so geared towards wealthy millennials who work in the tech industry, right?

Lea Alcantara: Yeah, yeah.

Aaron Gustafson: Like it's kind of gross.

Lea Alcantara: Right.

Aaron Gustafson: Yeah, but at the same time, there's really awesome stuff that's going on too, like I'm not to be like plugging Microsoft because I don't intend to like go around like toot horns on stuff like that, but I mean, I look at some of the accessibility stuff that's coming out and it makes me so proud to work in Microsoft because they at Build (Conference), which is like Microsoft's big developer conference, they showed a video of a blind man whose glasses could take a picture of what's in front of him and then tell him what it is that's in front of him.

Lea Alcantara: Oh wow!

Aaron Gustafson: Like that is freaking amazing, and that's where a lot of this machine learning stuff. I mean, you've seen probably some of the little demo apps that Microsoft has been putting out, showing the Machine Learning stuff, where it guesses your age or it can tell you what's in the photo or what have you, but a lot of this stuff is feeding back into creating these sorts of tools that will enable people who can't see the sea or to be able to at least know what's going on in front of them, and I've

seen some other examples of assistive glasses that can actually read to a blind user, like read an actual book to them with no Braille at all required, that just does kind of an OCR on the fly, which is amazing, and that stuff gets me really excited.

And I wish more people were working in that space because I do tend to feel that it's very easy to stay in your kind of comfortable myopic bubble and only build stuff for yourself. So Pew Research does a study every year of mobile stuff, and in the 2015 research (Chapter 1: A Portrait of Smartphone Ownership), there is a really weird finding that they discovered, which was, so with mobile users, smartphone users who fell in the $30,000-a-year and under income tax bracket in the US, they experienced app errors, I think it was either 53 or 54% of the time.

Emily Lewis: [Agrees]

Lea Alcantara: Wow!

Aaron Gustafson: Now, think about that, app errors. So this is an installed thing. They've downloaded it and installed this on their smartphone. Now, why are they experiencing errors more than half the time in using their apps? Well, it probably has something to do maybe with the network and availability of a network or the speed of the network.

Emily Lewis: [Agrees]

Aaron Gustafson: Or what is more likely, if I were to hazard a guess, I would guess that it's probably vastly different specs for what the developers are designing the site or the app rather for, especially like in the Android world, there's the flagship phones which have superfast processors and really high resolution screens.

Emily Lewis: Right.

Aaron Gustafson: And then you can get an Android phone that's a pay-as-you-go phone that's like $25 at Target from TracFone or something, and that's got specs like a smartphone from 2007.

Emily Lewis: Right.

Aaron Gustafson: So it's got a much slower processor. It doesn't have as much RAM. It doesn't have as much storage space. My guess would be that the fact is that if you're in a $30,000 and under tax bracket, you're probably not buying a flagship phone, but you're probably buying more of a lower end phone or an older phone that somebody had sold to getting that flagship phone, and that's why you're experiencing problems, and I think we're surrounded by the latest and greatest technology because we work on the web or we build apps and most of us have the luxury of being well above that $30,000 a year tax bracket, and are similarly surrounded by people who are in that scenario and so we fall into this trap of thinking that that's the experience that everybody has.

Emily Lewis: [Agrees]

Aaron Gustafson: And it's reinforced by movies and television where everybody has got high end phones. I mean, unless you're watching *Sons of Anarchy* where everybody is still in the flip phone.

Emily Lewis: [Laughs]

Lea Alcantara: [Laughs]

Aaron Gustafson: It's the reality on television is that everybody has got a smartphone and it usually looks like a high end Android or it may actually be an Apple device or something like that. So yeah, it's a comfortable place to be, but when we design for that experience, we're excluding all of the people who don't fit our lifestyle and our abilities, our income and all that sort of stuff.

Emily Lewis: I translate a lot of what you're saying as a cost to society to access to information, but it's also a real business cost for organizations and companies that are building or paying people to build these apps for them because they're cutting off potential market share. They're putting themselves in a position where they're going to have to mediate problems as opposed to building something from scratch and mediation always costs more.

Aaron Gustafson: Absolutely.

Emily Lewis: I can't think of a situation where it doesn't. Even in a house, if you have a problem and you have to like mediate a rate on problem, that's a lot more expensive than addressing it in the first place.

Aaron Gustafson: Absolutely.

Emily Lewis: But I feel like I wanted to make sure that we talk about some of the practical aspects of progressive enhancement so that we're not only trying to convey this mindset to our listeners, but also what they can do. So let's talk a little bit about how you approach progressive enhancement when you're building.

So when we first talked to you, and I still got this in the second edition, that they are like these building blocks of progressive enhancement, you've got your text, your semantic HTML, your CSS, JavaScripts, ARIA. So in terms of utilizing these different technologies to build something, how does a workflow look like for you?

***Timestamp**: 00:29:49*

Aaron Gustafson: I would say that a workflow for me varies by project, and actually, I'll talk about an interesting approach that we took on a project that we were building that I actually really like now,

which was when we did the project, we did the IA and had kind of all of buttoned down, and obviously, I had put some thought into how we were going to progressively enhance different interfaces, like this, is that calculator on this page and the calculator makes a round trip to the server?

If JavaScript is not available, then it goes back with the amount that the final total should be. But if JavaScript is available, then we do that as Ajax request or maybe we do it as just an in-JavaScript calculation rather than even making a trip to the server. So we kind of planned that stuff out ahead of time, but we needed to kind of expedite the project, the timeline was fairly aggressive, and so starting from the IA and the content strategy that we had done kind of part and parcel with that, the wireframe documentation and information architecture, I went about building out the HTML of the site first, thinking about all of the semantics and how the page should read and all of that good stuff, and then actually, we built the core functionality to work with our JavaScript, and then did the JavaScript portion of it, all in the absence of design.

Emily Lewis: [Agrees]

Lea Alcantara: Oh, interesting.

Aaron Gustafson: So all of this stuff that's going on while the designers were working on a visual design for the project, and then what that enabled to do is just kind of focus on what are the experiences and not letting visual design get in the way of it. In that way, I could think about when there is JavaScript available, what is this experience for somebody who is using this as a technology or screen reader, because no JavaScript is not a screen reader scenario.

In fact, there are many instances where we can use JavaScript to actually enhance the screen reader's experience or in assistive technology experience. It's not like the olden days where

JavaScript necessarily is going to get in the way of somebody being able to use a page. So that actually let me focus on that, and then circle back to make sure that the visual design and the JavaScript didn't undermine that core experience in any way, which was kind of cool. But I mean, in terms of building, it's all about thinking, "Okay, what is the baseline experience when it came time to work on the visual design?"

Because the JavaScript is pretty straightforward, either this JavaScript can work or it can't, so as long as there's a fallback for it not working or for it working without JavaScript, then we're good, and then, is the JavaScript experience still accessible? But on the design end of things, it's, "Okay, what's the baseline experience?" And in most cases, the baseline experience is for older devices that understand or older browsers that understand CSS, but only understand basic CSS, maybe they don't understand media queries, maybe they don't understand advanced positioning techniques and stuff like that.

So we can provide a linear design experience, so maybe it's just typography, color, maybe a little bit of like borders and background colors and stuff like that to create some visual differentiation, but it's a single column view, and then that becomes the baseline style sheet that's delivered universally, and then there's an advanced style sheet that has a media query as part of the link tag, and so that media query will actually preclude that style sheet from being downloaded in any browser that doesn't understand media queries because they won't know what to do with that media assignment so they'll just basically skip it, they'll ignore it, whereas modern browsers that do understand media queries will also download that additional style sheet, so that enables us to them sequester all of the rules related to positioning and layout and all of that sort of stuff within that media query associated style sheet, and then we can do sub-media queries inside of there.

So maybe that media query style sheet is screen or rather only screen, you could just say, which is a simple media query, and you're not even doing min-width or anything like that. You're just saying only screen and that only keyword will basically cause it to be ignored by any older browsers, so then within that only screen style sheet, you could do your media queries for min-width 320 or min-width whatever 20ems 64ems or whatever your breakpoints are to build up that experience, and as long as you're starting from that mobile-first perspective and building up the design in your CSS, you're creating a progressively-enhanced design.

And then if you want to take things a step further, we now have @ support for being able to test whether a particular property is available or particular value for that property is available in the browser that we're in, and then only applying rules within that block to those browsers to further enhance things.

So it's just kind of creating a step-up approach and if the browser passes this capacity, then do this. If it has this capacity, do that, and just kind of approaching things in a methodical way to create those enhancements.

Emily Lewis: With the approach you just described, is there any argument against having an additional style sheet called from a performance perspective, which also feels like it should be considered with mobile-first, so having more than one request?

Aaron Gustafson: Yeah, I mean, there are a couple of different ways that you could go about doing it. You could certainly embed that initial mobile style sheet first.

Emily Lewis: [Agrees]

Aaron Gustafson: You could use kind of the critical CSS approach, doing that if you've got something that can be more automated. I would say it comes down to how big your style sheets are and thinking about it from the perspective of those older devices.

Emily Lewis: [Agrees]

Aaron Gustafson: So if an older device, let's say it's a – I don't know – a Trio 650 as somebody might still be using one of those, or let's say an older BlackBerry device, because there are still BlackBerry 5s around, for instance, and those are not WebKit-based or anything like that, so it's really like old and they have a hard time with a lot of modern stuff, so if we're giving them a mobile style sheet and then there's an advanced style sheet, they'll ignore that advanced style sheet because it's got a media query and they have no idea what to do with that, so they just download that single, very, very minimal style sheet so they have one request.

In more modern browser, it's going to have to make two requests, but I mean, you can do things like resource hints to basically pre-fetch the DNS lookups and all of that good stuff to optimize those request from modern browsers so that it doesn't become an issue.

Emily Lewis: [Agrees]

Aaron Gustafson: I mean, we haven't seen that much of a performance impact, and if you've got a lot of styles that you're dealing with, if it's a really big site, being able to sequester all of your layout in advanced style sheet stuff within one style sheet and delivering a really slim style sheet to older devices can be a really good approach.

Emily Lewis: Yeah, I feel like that scenario is a good example for why developers need to understand the fundamentals of what they're doing.

Lea Alcantara: Right.

Emily Lewis: Because it's a project by project or scenario by scenario basis. You need to understand what you're building, and how it's being called into the browser and how everything affects everything so that you can make those judgment calls. You can decide. If I'm going to build this way and I'm going to utilize more than one, let's say, style sheet per asset, I'm making a cognizant choice that that's actually more valuable to me than that additional asset that might be called, and it only affects performance this much, and just being aware of what's happening so you can make informed decisions.

Aaron Gustafson: Absolutely, it's always about tradeoffs.

Emily Lewis: Is that clear? I remember back in my early days getting into web standards, I was all about the rules. [Laughs]

Lea Alcantara: [Laughs]

Emily Lewis: And now I just understand that the rules are a guide to help me make a decision, that it's not about always following all the rules to a T, it's having the knowledge and experience to know what the rules are and know why I might choose to not follow one.

Aaron Gustafson: Exactly.

Emily Lewis: So we actually got a couple of listener questions that are pretty specific about implementation. I'm not going to throw all of them at you because otherwise it will be like a workshop tutorial, but let me see what your thoughts are on this question from Kevin Lozandier. He asked, "What is your advice on achieving basic accessibility without JavaScript when you have widgets that require state and keyboard shortcuts for their core functionality?"

Aaron Gustafson: When we're looking at an interface today, let's say you're building a chat application, an outline chat, as though we need more of those.

Emily Lewis: [Laughs]

Lea Alcantara: [Laughs]

Aaron Gustafson: But you're building a chat app and you may think in your mind, "Oh, I need web sockets in order for this to be able to work, because web sockets keep a channel open for communication back and forth between the server and I can make sure that I can get information across or I need to push notifications or I need this, that or the other." And yes, but no, right?

Emily Lewis: [Agrees]

Aaron Gustafson: Like none of the things that we're building today are new challenges, right?

Emily Lewis: [Agrees]

Aaron Gustafson: Like these are things, like we had chat in the web 1.0 days.

Emily Lewis: Yeah.

Aaron Gustafson: We had to do a meta refresh and it did a round trip to the server and refreshed the page every time or every time you hit the send button, but we had chat. It kind of sucked, but it worked, right?

Emily Lewis: [Laughs]

Aaron Gustafson: And you can have that as a baseline experience even if it's kind of a shitty experience, pardon my French, but you can have that experience and then if there's a better experience you can provide based on the capability, it's like based on having web sockets or based

on having push or whatever, you can then override that baseline experience and hijack it, but I think a lot of it comes back to looking at how we built this stuff in the very early days of the web and thinking about that as kind of the baseline experience.

We used to do round trips and make requests for a new page and you navigated to that new page, and that new page had content and we had dashboards and we had all of these other things that can completely be done and work for anybody and then we can take that over and make far more interactive experiences and experiences that don't require as much round trip to the server or it only makes smaller request in that sort of stuff.

So I think it's all about thinking about what is the simplest way that I can accomplish this goal, enabling somebody to do what it is that they need or want to do with the simplest way I can achieve that for them and then or at least, least technically complicated and then how can I make that better.

***Timestamp**: 00:40:47*

Emily Lewis: [Agrees]

Aaron Gustafson: And in a lot of cases, I think developers excuse themselves from having to do that or take that approach because they're like, "Oh, then I have to write the code twice and blah, blah, blah." But not really, because in most cases, most developers have gotten on board with the idea of the APIs and if you're building an API-based architecture for your site, your product, your app, whatever it is you want to call it, then there's no reason that the server side can consume that same API that the client site would be consuming, so you can minimize the amount of code duplication or if you're going on it on JavaScript, you can take the isomorphic JavaScript approach and execute your

JavaScript, your client side JavaScript code on the server side and Node.js or something like that. So it's possible to have your cake and eat it too.

Lea Alcantara: Well, I do feel like the follow up question to that, Kevin also asked, which might put a wrench in the stuff that you just said, he was mentioning offline-first way of making apps becoming increasingly popular, so some of the fallbacks that you mentioned was like, "Okay, well then the server will pick up the slack here," but if you're developing something offline, it looks like there seems to be a push amongst developers to deliver experiences that can't be obtained without JavaScript. What's your perspective on that?

Aaron Gustafson: It's funny I actually just wrote a blog post about that the other day.

Lea Alcantara: Oh.

Aaron Gustafson: I love the idea of "offline-first," but I feel like as a name, I have issues with it.

Emily Lewis: [Laughs]

Aaron Gustafson: Because it masks the fact that offline is never first. Unless you're talking about something that is a preinstalled application, the network is always required to deliver it, right?

Lea Alcantara: [Agrees]

Aaron Gustafson: Nobody is delivering their websites on USB keys and CDs anymore.

Emily Lewis: Right. [Laughs]

Lea Alcantara: Right, right.

Aaron Gustafson: So the network always comes, well, not always comes first. In IOT, the network won't come first because you're dealing with an embedded thing, right?

Lea Alcantara: Right.

Aaron Gustafson: But "Offline Early," yes, absolutely, and if you have an experience that works without JavaScript and works in kind of that old way, that will work for anybody who doesn't have Service Worker yet or those sorts of things, and if you do have Service Worker, you can go ahead and you can offline a bunch of the assets and make sure that pages are made available and so on and so forth and absolutely take advantage of that, but you're not precluding somebody from being able to access your site because they don't have Service Worker. I mean, that's kind of the brilliance of Service Worker is that it is a progressive enhancement.

Emily Lewis: [Agrees]

Lea Alcantara: [Agrees]

Aaron Gustafson: You do a test to see whether the browser supports it, and if it does, then you go ahead and you load it in that Service Worker and it takes care of handling the request. But if you don't, then the browser is none the wiser. It just goes ahead and carries on the way that it normally would. So I mean, that's why I've implemented Service Worker on my site, my blog, and I think offline is a really important thing, a really important scenario to consider because things happened.

Emily Lewis: [Agrees]

Aaron Gustafson: People move through tunnels. People end up on hotel Wi-Fi that can sometimes be a lot slower than some of the worst mobile connections. There are lots of instances where the network is unreliable or flaky, and absolutely, we should do everything that we can to create experiences for our users regardless of whether the network is available, but I think the offline-first name, it sort of makes you think that offline can happen in the absence of the network, and there are

very few scenarios that it can, and in reality, you've got two dependencies, first, you have the network if you're talking about a website, and then you have JavaScript.

Emily Lewis: [Agrees]

Aaron Gustafson: And so if neither of those two were available, there's no offline.

Emily Lewis: [Agrees]

Aaron Gustafson: It's offline-never, right? [Laughs]

Emily Lewis: [Laughs]

Lea Alcantara: Right.

Aaron Gustafson: You're effectively dead in the water. So I'm absolutely in favor of offline and planning for that as early as possible, but I think you also have to have a plan for when that capability is not available.

Emily Lewis: [Agrees]

Aaron Gustafson: And hopefully, as Service Worker becomes better defined and more browsers begin implementing it, that will become a reliable approach for future web designers at least in modern browsers, but I think we'll always be dealing with straggler browsers. We've always been dealing with straggler browsers.

Emily Lewis: [Agrees]

Aaron Gustafson: So having a scenario for those people too is, I would say, every bit as important, if not more important.

Emily Lewis: So I'd like to take the conversation a little bit to responsive web design and mobile design because, for one, you mentioned your second edition has a much more deep discussion about that topic compared to the first edition, and so much has happened in those five years. So what do you feel now that you've had the time to reflect on it since the first book and you've gotten your hands even more into responsive design, where does it align with adaptive web design? What are the things that you have to take into account when you're building something that needs to be responsive, but it also needs to be progressively enhanced?

Aaron Gustafson: So I think the sweet spot is responsive design from a mobile-first perspective because if you're doing desktop-first, which a lot of people have kind of used as a way to shoehorn responsive design into an existing website…

Emily Lewis: And Aaron, before you go on with that…

Aaron Gustafson: Sure.

Emily Lewis: When you say desktop-first, you're talking like the front-end developer has built all the styles for the desktop experience and then they build the styles to kind of undo for mobile?

Aaron Gustafson: Yes, absolutely.

Emily Lewis: Okay.

Aaron Gustafson: And so what you end up with is you end up creating a scenario where every user is downloading all of those desktop styles whether they're applicable or not.

Emily Lewis: [Agrees]

Aaron Gustafson: And so the desktop-first approach is very much kind of a subtractive or reductive approach, and so you're having to write more code to undo positioning or undo floats in order to bring things into single columns or what have you, and so you end up writing almost twice as much as code to undo stuff so you're creating more bloat in your CSS in addition to forcing users on older devices to download stuff that they may not use, and if a browser doesn't understand media queries, it's going to get that desktop view.

Emily Lewis: [Agrees]

Aaron Gustafson: So I feel like there are a lot of downsides to that approach, and yet that is still technically responsive design, if you're following up with grid and all of that stuff.

Emily Lewis: Right.

Aaron Gustafson: So taking it from a mobile-first perspective, you're practicing progressive enhancement. You are building from a mobile, single column, low capability view and then as the browser is capable, as in has more screen real estate or what have you, you are enhancing that display in order to provide a better reading experience or have a little bit better visual distinction between elements to guide people's eyes to the page in slightly more efficient ways and so on and so forth. You're able to adjust hierarchy and et cetera, et cetera.

But the other thing that I really like about the mobile-first responsive approach is that it keeps your focus on the primary purpose of the site. As web design progress and our monitors got bigger and bigger, we instinctively started to fill that space with stuff that was a distraction from the primary purpose of every page, right?

Emily Lewis: [Agrees]

Aaron Gustafson: I feel like the mobile-first approach really keeps us honest, and when Luke Wroblewski coined that term, it was all about putting a focus on what the user is there to do on that given page and eliminating all of the excess noise, and I feel like the responsive design approach from mobile-first really picks up on that and then enhances that experience based on the amount of scree real estate you have available to you.

Emily Lewis: [Agrees]

Lea Alcantara: So I have a question regarding that because in your earlier discussion over how you've incorporated progressive enhancement in your current workflows, you actually start developing the site very, very early before the design is even completed, so then when does the design fit in? Like at what point does design start to effect progressive enhancement, and what are the things that designers themselves need to think about while they're designing their comps and their style guides and all those kinds of things in order to fit in with progressive enhancement?

Aaron Gustafson: I mean, I think one of the things that we've seen the most beneficial is to actually take a component-based approach to design, so working to create a style guide or working to create a pattern library, and so the information architecture, the wireframes, identify all of the different components that the designer is going to need to work with, and then if the designer is working from that as kind of a laundry list of what they need to think about, the IA can also kind of illustrate to them what are the different experiences of this interface. Is this navigation menu an off-screen navigation menu that slides in? And then what are the different interactions? What are the different endpoints? So I've written a bit about this.

***Timestamp**: 00:49:55*

There's an approach I like I use, which is basically just flow charts, but I refer to them as interface experience maps where you're actually mapping out what are the decision points as the interface is being created, and those decision points are based on like JavaScript/no JavaScript, is this API available on the browser? Does the browser support this? Is the browser wide enough?

That sort of stuff, and it actually identifies, what are the different experiences that a user may have of this same interface, and so using that approach and kind of outlining, here are the different experiences of this interface, a designer can then look at that and say, "Okay, I need to make sure that I have designs that accommodate each of these scenarios," and that goes over to the front-end developer rather to the integration and it goes to the QA team because then they can look at that same IX map and say, "Okay, I know I should experience these different things in these different scenarios," and they can actually test to make sure that they're getting that experience.

So that becomes a really helpful tool, and I've kind of walked through that as well in the final chapter of the book, how you can take that IX map approach to different interfaces, and one of the clients that I worked with actually, they kind of made their own version of Pattern Lab that for each component, they included an IX map for it within Pattern Lab or within their port of Pattern Lab, but they also created a dropdown on the site of the overall pattern library interface that allowed them to toggle between no JavaScript/no CSS, no JavaScript/basic CSS, no JavaScript/advanced CSS and JavaScript/advanced CSS, and so they could actually toggle each interface that they were viewing within that pattern library in each of those scenarios, and that they could use them to test and then obviously squish and enlarge the viewport and all of that sort of stuff, which was really cool, and so I find that that approach to design where you're taking kind of a more methodical approach to designing it and designing all of these different pieces and different experiences of those pieces ends

up creating a better overall experience and a more consistent experience for the end user than designing each page specifically, which can become quickly overwhelming.

Emily Lewis: Yeah, I love that. I really do. I think the idea of mapping out the scenarios, I can't think of how many times in the past, and I'm not talking about you, Lea. [Laughs]

Lea Alcantara: [Laughs]

Emily Lewis: Working with designers where I don't get design direction on when JavaScript is not enabled, and so I then not only am developing and testing, I'm having to design in the browser, which isn't a problem, I'm capable of it, but does the project have time for that? And by the time it gets to me, there's this much time, and I just feel like that not only is going to help the designer and their communication with the developer, but I love how you then take it to the QA phase. It gets so much easier for whoever is in charge to test, to know what they have to test for.

Aaron Gustafson: Absolutely, and it's nice because they're flow charts, they're very easy to do on white boards, do on paper, do in brainstorming sessions. Anybody can be involved in the kind of creation process, whether it's the product manager, somebody who works in content strategy and so on and so forth, folks that aren't as schooled in the ins and outs of CSS and HTML and even visual design can participate in the discussion of what those experiences are, and then each of the people who has a deliverable that's part of creating that experience knows that they have a part to play in the creation of those experiences.

Lea Alcantara: [Agrees]

Aaron Gustafson: So from a content strategy standpoint or a copywriter standpoint, you know that like you may have slightly different content that you need to author for this scenario over this

scenario, and so you can take that into account early and you don't end up with being three days from launch and realizing, "Oh crap, we need to write copy for this alternate experience."

Emily Lewis: I love it. I do. I really just think that's such a useful tool to a project and how it can affect all the different phases of a project. I think it would even be great, you could even share with the client so that they could even see like, "Oh, and we've covered all of these scenarios as well," just to further illustrate the value of what you've built for them.

Aaron Gustafson: Absolutely.

Emily Lewis: So we just talked a little bit about QA, let's talk a little bit about testing in general. What is your process to test, to cover as many bases as possible when you're building something? I mean, and this goes beyond just like having a lot of devices even, right?

Aaron Gustafson: Right. So I mean, I'll admit that in most cases, I'm not doing a ton of testing during the development phase.

Emily Lewis: [Agrees]

Aaron Gustafson: And the reason for that is that I've been doing this for so freaking long that I have a lot of embodied knowledge that I know a lot of the quirks and intricacies of the way different browsers work, but at the same time, I also am happy to draw certain lines in the sand for certain browsers. So for instance, just like I talked about using media queries as a way to keep older browsers from getting modern CSS, I really like the approach of inverting conditional comments for older versions of IE, which tend to be, at least in terms of desktop, like the older browsers that we're dealing with and you can invert a conditional comment such that older versions of IE do not get anything that's contained within them, but all modern browsers get them.

Emily Lewis: [Agrees]

Aaron Gustafson: So I use that approach to actually keep IE 6, 7, and 8 and maybe even 9 in some cases from getting JavaScript delivered to them at all and so they get the non-JavaScript experience and that eliminates my need to test on those browsers.

Emily Lewis: [Agrees]

Aaron Gustafson: But because I have a no JavaScript fallback, I know that users on those browsers that they happen to be stuck in IE 8 because they can't upgrade from Windows XP for one reason or another, they can still have a decent experience, a positive experience and do what they need to do on the site, but I can focus my browser testing on more modern browsers and making sure things work there.

Emily Lewis: [Agrees]

Aaron Gustafson: I mean, obviously, I'm testing JavaScript as I'm writing it and testing CSS as I'm writing, but in terms of device testing, in a lot of cases, that ends up kind of coming towards the tail end, but in most cases, I nip here or tuck there, and I mean, progressive enhancement as an approach, it just works, and to kind of illustrate that, there were two projects that we worked on around about the same time. We had one project that we did that was a Chrome app for WikiHow and this was when Chrome apps were just coming out, and so we built it as a Chrome app. We did it based on what Safari was doing at the time because Chrome at that time didn't support things like 3D transitions or 3D transforms rather. So we built it according to what Safari supported knowing that Chrome was going to pick that up in Chromium and then that would be appearing by the time the store rolled out and this app rolled out, but we built it only as a Chrome app.

Emily Lewis: [Agrees]

Aaron Gustafson: It was a website, but it was made or tailored for Chrome so we did put up a block page if you did try and reach that URL in another browser. I think we let Chrome and Safari through, but that was it. It's a WebKit only. And the reason for that is that's what the client requested and they wanted to use a lot of cutting-edge and I'm using air quotes here, "HTML 5" stuff that included like WebDB and such, which was SQLite in the browser, which is now an abandoned spec.

So we built it for that and delivered that project, and let's say that cost X dollars, they came back to us six months later and wanted us to make it work in the then current version of Firefox and IE, which were they're quickly Firefox 14 and IE 9, and IE 9 didn't support transitions at all at that point and didn't support some of the other stuff, and neither Firefox nor IE supported the SQLite in the browser, the browser side database, and the reason for that was that neither of them wanted to implement SQLite within the browser. They didn't want to have a standard based on a third-party piece of software.

Emily Lewis: [Agrees]

Aaron Gustafson: And so neither of them implemented it. The replacement for it, IndexedDB, was still being finalized by the W3C and so no browser had an implementation of that, so we had to figure out how to do a client side data store that would work for all of these different browsers, so I effectively ended up rebuilding the API we have for Web SQL so that it would work with local storage and I used proprietary user data in older versions of IE so that we could have that.

So I built a wrapper that enabled us to basically port our code over in storage separately in local storage as key value pairs and stuff. But anyway, so porting that project from just being Chrome or

Chrome and Safari over to supporting two new browsers, we estimated it as costing 40% of the original budget.

Emily Lewis: Wow!

Lea Alcantara: Wow!

Aaron Gustafson: It actually ended up coming in a little bit over that.

Emily Lewis: Yeah, I'm not surprised. [Laughs]

Aaron Gustafson: So that was just adding two browsers, right?

Emily Lewis: [Agrees]

Aaron Gustafson: Now, on the flipside, there is another project that we did for a large investment company where we were targeting only their apps. They were going to open the website that we were creating for them which handled their login flow only within their iPhone, iPad and Android apps, and then maybe their Windows Phone app if they developed one.

They weren't sure if they're going to develop one or not. So we were only testing on those four platforms, and so let's say that project cost Y, we built that using progressive enhancement because that's how we build things. If nobody is asking for otherwise or giving us a valid reason otherwise, so we built it all in progressive enhancement, delivered them a pattern library and they kind of built out everything, we did like 35 representative screens and made like a 100 and some odd that they took and built out from there, but everything worked without JavaScript and with JavaScript and we made it so that it worked from small screens up to large screens, and even though we knew we were only dealing with full screen on iPhone and small Android devices, but then like windowed within iPad and

such, and so they came back to us about six months later and said, “Hey, this has been great. We want to roll this out to our m. site. Here’s our list of 1,400 user agent streams that accessed login in two days. We need to be able to support this.” And that was a really huge spreadsheet.

***Timestamp**: 01:00:27*

Emily Lewis: [Agrees]

Aaron Gustafson: And when you look at 1,400 devices, you’re like, “Oh my God, this is insane. How are we going to handle this?” So I wrote an algorithm that would basically parses user agent strings and gather data from an online resource in order to help us see what those user agent strings we’re referring to. In other words, what’s the OS? What’s the browser? What’s the version of each of those? How much screen size are we talking about? What are the capabilities, et cetera, et cetera, et cetera, so that we could begin aggregating those into manageable chuncks.

Emily Lewis: [Agrees]

Aaron Gustafson: And so we managed to bring that down to about 23 devices that would be representative of about 97% of the spectrum.

Lea Alcantara: [Agrees]

Aaron Gustafson: There were some that we just could not find a device for because the browser was only on devices that didn’t support Wi-Fi, for instance, and we didn’t want to have to deal with getting a data plan just to be able to test on that device, and in some cases, we have to fall back to emulators, but on that list were like BlackBerry 4, BlackBerry 5, both of which were pre-WebKit. Some really nasty stuff, none of which supported HTML 5 input types and all that sort of stuff. So we gave them an estimate of 30% of the original budget in order to add these 1,400 new devices, and

then when we actually began testing and making tweaks to what we had created, we actually came in at half time and a half budget.

Emily Lewis: Wow, and that's because of progressive enhancement?

Aaron Gustafson: Absolutely, and as far as I know, that had never happened in a project that they've had before. They didn't even know how to take money back from us. [Laughs]

Emily Lewis: [Laughs]

Lea Alcantara: [Laughs]

Aaron Gustafson: So we ended up looking great to our client. They ended up looking great to their boss and so on and so forth, and it was all because this approach just works.

Emily Lewis: [Agrees]

Aaron Gustafson: I mean, it works really well for enabling you to support a wide range of devices. So here, for effectively, 15% over the original budget, we were able to add 1,400 new devices as opposed to upwards of 40% over the original budget to add to.

Emily Lewis: So I think that case study or scenario with the client is a good selling tool for progressive enhancement. Before you were at Microsoft, you had your own shop, so is that something that would be part of your sales process or even after you've gotten the project, communicating with the client the importance of following a certain approach, or is that something that just happens behind the scenes?

Aaron Gustafson: In a lot of cases, it just happened behind the scenes, and that was the way it was in previous positions that I've had. In terms of accessibility stuff, it was just something that I did or

that I had our team do, and had us incorporate and then it became a kind of a surprise to our client, like we had a state and government agency that we had done a project for at a previous agency that I had worked at, and they came to us literally the day after the site launched and they were like, “Oh my gosh, we have to meet Section 508,” and I was able to say to my boss, “Well, it already does because that’s what I do.”

Emily Lewis: [Laughs]

Aaron Gustafson: And so in a lot of cases, it goes on behind the scenes. As Easy Designs, as a consultancy kind of grew, it went from being a behind-the-scenes thing to a reason that clients were actually coming to us so we didn’t actually need to do the sales pitch because they knew us for doing that and so they would come to us because we used that approach and they knew that their stuff was going to work no matter what.

Emily Lewis: So I feel like that’s kind of a nice segue way to another question that Kevin sent. So do you have any advice for developers who are advocating for accessibility at their workplace, even citing like legal reasons, but they get feedback from stockholders, their bosses, that they’re not going to bother with it until they’re sued?

Aaron Gustafson: Such a positive attitude. [Laughs]

Emily Lewis: [Laughs]

Lea Alcantara: [Laughs]

Aaron Gustafson: No, I mean, I had a coworker at a previous agency that when the Target lawsuit was going on, he was saying that basically he didn’t think that Target should be able to be sued, that they should be able to exclude anybody they want from the thing.

Emily Lewis: What?

Aaron Gustafson: And I've had clients that, or not clients, direct clients, but people I've heard of saying things like, "We sell TV because blind people don't buy TVs."

Emily Lewis: Oh my God.

Lea Alcantara: [Agrees]

Aaron Gustafson: Which is so incredibly misinformed, but I mean, I think there's a great presentation on inclusive design that comes from the Microsoft design team, and then part of the talk, they talked about sort of the continuum from a permanent disability to a situational disability.

Emily Lewis: [Agrees]

Aaron Gustafson: So if you think about somebody who is missing an upper extremity, like they're missing their right arm, that is a disability in the classic sense and there may only be – I can't remember the statistics specifically, but let's say it's like 7,000 people that lose an upper extremity in the US every year, that's permanent, and then there's temporary, which is maybe somebody broke their dominant arm and so they're having to not use a keyboard or maybe they're mousing with their non-dominant arm, which is not as accurate, so they have a temporary disability, and there may be somebody who is an avid mountain climber, and if you have a store that sells mountain-climbing gear, you might be like, "Oh, we don't need to cater to people that only have one arm."

Emily Lewis: Right.

Aaron Gustafson: But there is that other scenario where somebody has a broken arm, or to take it to a situational dependency that I've come to understand being a new parent. Being a parent of a

newborn child, there are many instances where you're holding the child and trying to do something else at the same time.

Emily Lewis: [Agrees]

Aaron Gustafson: And that only happens at certain times of the day or at different times throughout the day so that's very situational. The same thing could be said for somebody who's on a mobile device. They may not always be on a mobile device. It may just be this situation that they're in at that time, and so if you begin to think about this continuum, basically the calculation that they made in this presentation was that, that one scenario of missing arm, broken arm, new parent, that's 21 million people in the US every year.

Emily Lewis: [Agrees]

Lea Alcantara: Right.

Aaron Gustafson: That's a lot of potential people that you might be excluding if you're not taking into account that disabled experience.

Emily Lewis: Right. And I have a further comment in response to Kevin's question that it's not just the lack of or concern of being sued, it's like you just said, that is a potential customer.

Aaron Gustafson: Absolutely.

Emily Lewis: It's not even an issue of what if that customer sues me. If that customer is not going to buy from you, you're not making money.

Aaron Gustafson: And they're not going to tell their friends about your product or they might tell your friends that your site sucks, right?

Emily Lewis: [Agrees]

Lea Alcantara: Right.

Aaron Gustafson: Actually, you could turn them into an advocate for you or you could turn them into a detractor.

Emily Lewis: [Agrees]

Aaron Gustafson: We built a site years ago, and this was let's say 2003, for a restaurant chain predominantly in the northeast, and this is another agency that I was at, and when we built the site, I built it with web standards using CSS and it was very accessible including a menu that wasn't a PDF.

Emily Lewis: [Laughs]

Lea Alcantara: [Laughs]

Aaron Gustafson: And the site originally had no Flash, and that was very much against the grain for that particular period of time on the web, and honestly, still is in restaurant world, you see still a lot of Flash stuff, I mean, certainly a lot of PDFs, and they actually got an email from a blind customer who said, "Your site is amazing. I wish every restaurant website was like your site."

Emily Lewis: [Agrees]

Lea Alcantara: Right.

Aaron Gustafson: You can't buy that sort of goodwill.

Emily Lewis: [Agrees]

Aaron Gustafson: And you can bet that that person was a staunch advocate for their restaurant chain and was probably talking them up to a lot of people. Of course, two years later they redesigned with another agency and it went all the Flash, you know? [Laughs]

Emily Lewis: Right. [Laughs]

Lea Alcantara: [Laughs]

Aaron Gustafson: But for a brief moment in time…

Emily Lewis: I think that even illustrate the definition of a rich experience is not the same to every person, you know?

Aaron Gustafson: Absolutely.

Emily Lewis: That person got a very rich experience that perhaps a sighted user might not appreciate because maybe it doesn't have the flash bang that they were looking for literally, but that was still a very rich experience for someone because their fundamental goal was to get information, and they got it.

Aaron Gustafson: Yeah.

Emily Lewis: And they got it in an easy way without having to struggle. That's rich.

Aaron Gustafson: Yeah, exactly. I often think about Mat Marquis' tweet, which was, "Mobile users want to get access to your menu, your hours and your locations, but desktop users really want to download that 20K image of a person smiling at a salad." Right?

Lea Alcantara: Yeah. [Laughs]

Aaron Gustafson: And that's really, it's the tension I guess between kind of the marketing and design aspects of a business and the actual usability and user experience.

Emily Lewis: [Agrees]

Aaron Gustafson: And there is definitely a tension between the two and I feel, like within a lot of design teams, there is this urge to create something that can be "award-winning" website, something that they can use in their portfolio and show off their design skills.

Emily Lewis: [Agrees]

Aaron Gustafson: But ultimately, are we creating websites to enhance our portfolio, or are we creating websites to solve our user's needs?

Emily Lewis: And I feel like, even taking that a slight step further, is that we often are in situations with our clients and someone found something they saw on a site that they just really want on their site because they really like it, and our job isn't just to build them what they want, but our job is to inform them of how that new flashy thing they're going to implement is not only going to take time, but how are we going to build that to address their entire customer base, and educating them that it's not just this cool new thing, that there are so much more involved, and even something as simple as we have a client who wants this massive image gallery kind of thing.

***Timestamp**: 01:10:23*

Lea Alcantara: Right.

Emily Lewis: And it's educating them and being like, "Yeah, I recognize that you really like the look of that, but can you imagine that on your phone? What would that be like for you?" And pushing

back, that is our job to get that client the solution that they've paid for, and that doesn't always mean giving them everything they think they want. It's advising them of the bigger picture and the implications for everything from performance to accessibility to the cost and their ability to reach their audience.

Aaron Gustafson: Absolutely, I mean, they're hiring us. In most instances, I would say they're hiring us for our expertise.

Emily Lewis: [Agrees]

Aaron Gustafson: And if we're not applying our expertise to their project, then we're not doing our job right.

Lea Alcantara: Yeah.

Aaron Gustafson: We're doing them a disservice, and I think in that scenario, if they see something that they liked, we need to talk to them and address that from a standpoint of, "Okay, what is that accomplishing in your mind? What is that doing for your users, and how is that helping your users accomplish their goals with regard to your site?" And then let's figure out, is that the most appropriate approach for your site, or is there something that we can do that will still meet those goals while being appropriate to your scenario?

Emily Lewis: [Agrees]

Aaron Gustafson: Because the site that you're looking at, they designed that for their scenario and their problems, their challenges, their end user goals and stuff like that are different than yours.

Emily Lewis: [Agrees]

Aaron Gustafson: I mean, that's the reason that I love Bootstrap as a learning tool, but I don't that every site should be a Bootstrap site because it was designed as a pattern library originally to solve the prototyping needs to Twitter.

Emily Lewis: [Agrees]

Aaron Gustafson: It's not necessarily designed to solve the needs of your site, and so I'm a big fan of building bespoke for every project because we don't all have the same requirements, the same needs, or the same problems to solve.

Emily Lewis: It just occurred to me that there is a lot of skill sets that you need to have to build something progressively, and one that I think just occurred to me is that you need to be able to talk to your clients about real business things, like the actual impact, numbers even.

You almost have to quantify things in terms of, like we doing some performance optimization for a client, and we were like, "Well, we'll make it better," and they're like, "Well, tell us how you're going to make it better," and so we took the time to map out charts and show them that if we cut the response times by this, this is what would happen, and these are your competitors are doing, and so it's less about talking to them about, "Oh, well, it's going to be accessible and it's going to responsive."

And using these terms that frankly the clients we talked to don't have a lot of connection with and putting it in terms of, "Well, your competitors are doing this, and this is where you rank with your competitors and this is how much market share you might be missing out if you're addressing this audience." So having a skill set to talk from a business perspective is hugely important when you are trying to not necessarily get buy in for progressive enhancement, but build things in a progressively-enhanced way when the client is asking for things that may not fit in with your initial plans to do so.

Aaron Gustafson: Absolutely, that was actually one of the things that I loved about Mike Monteiro's *Design is a Job* is there is a whole section in there where he basically said, "If you're a designer, you need to be able to walk a client through your design and explain why things are the way they are in design, the rationale behind it, and not, 'Here you see this design, and in the upper right hand corner, we did an orange button, and here we have the main text,' like that's not helpful if we talk about why you chose to put things in a particular place and relate that in a 'Here are your business goals for this particular page.

Here's why we designed it the way that we did, and so on and so forth." And absolutely, I think that web designers in terms of like front-end designers and developers and such need to be able to have that same level of expertise to be able to have those conversations, not talking about specific technologies that they're using, but what are the goals in terms of metrics like you're talking about and are there ways that we can address those, and are there tradeoffs too if we do this.

Emily Lewis: [Agrees]

Aaron Gustafson: This is completely possible to do, but do we want to go that route.

Lea Alcantara: Right.

Aaron Gustafson: So an example, we had a client that, the one that we were working on the mobile apps web stuff for, asked us about doing a toggle control rather than a checkbox, and so we went down that path of saying, "You know, we can absolutely do that, it's technically possible to do, but let's do a little exercise and think about this. So if we create a custom toggle control, obviously, there are CSS and JavaScript that needs to go along with that, but then we have to decide what does the design would look like. Do we match that design to iOS?" And at that time, we were doing iOS 6.

When iOS 7 comes out and it has a vastly different design, do we change all of them to look like iOS 7 or do we now maintain two different designs that map to those two different things? What do we do about Android? What do we do about the proliferation of different skins for Android? Like do we want to go down this rabbit hole basically?

Emily Lewis: [Agrees]

Aaron Gustafson: Like it's technically possible to do, but is it worth it?

Lea Alcantara: Right.

Aaron Gustafson: Is it worth it from a development standpoint and on money standpoint initially, but also, is it worth having to maintain that code down the line, and are you getting enough value out of that to actually make it worthwhile? And I think those are the conversations we need to be having with our customers, and in many cases, they're going to say, "No, let's just use the checkbox." [Laughs]

Emily Lewis: [Laughs]

Lea Alcantara: [Laughs]

Aaron Gustafson: People know what a checkbox is, it's okay, but I think that's our role and I think that there are not enough web designers and developers out there who are embracing that role.

Emily Lewis: Yeah, and that honestly I feel like it's its own podcast sort of in and of itself. [Laughs]

Lea Alcantara: Absolutely. So our loyal listener, Kevin, who's been asking us all these great questions to send you, I think has a really good final question, "So what's your advice on developers

who continually want to better understand accessibility and internationalization and want to consider themselves qualified, but have no idea how to definitively evaluate this?"

Aaron Gustafson: I mean, I think the first thing is just read, read, read.

Emily Lewis: [Agrees]

Aaron Gustafson: I mean, there are a lot of high-quality blogs and white papers and stuff like that, I mean, with information that's being put out by organizations like Deque or Paciello Group. The W3C is getting better at writing more developer-friendly implementation guides for things WCAG, which is the Web Content Accessibility Guidelines. There are a lot out there to learn, and I think always being open to learning new things is super important.

Emily Lewis: [Agrees]

Aaron Gustafson: But I think the other part of that is recognizing that nothing is ever going to be perfect.

Emily Lewis: Yeah.

Aaron Gustafson: It's always a process, and as we learn, we get better and we can go back and refine things.

Emily Lewis: [Agrees]

Aaron Gustafson: I mean, that's why I love pattern library because it gives us the opportunity to look at a specific interface component and iterate on that one component in isolation to make it better and better each time, and if you can find a way, like Lonely Planet did, for instance, to connect your live website to your pattern library when you make those updates to improve the accessibility of an

interface based on new things that you're learning, that's automatically filtering out to your web products and improving them immediately such that you're not running into an issue where somebody had to manually use that component in a design that they were doing and so it's disconnected from your style guide and end up being out of sync.

So I think any opportunities to do that and to create that direct connection between your live site and your pattern library or your style guide is definitely an awesome thing to do, and then iterate and just keep learning and keep figuring out newer and better ways to do things and test.

Emily Lewis: [Agrees]

Aaron Gustafson: Test with real people. See how things work. If you learn something, share it.

Emily Lewis: [Agrees]

Aaron Gustafson: Give talks whether it's to a local Meetup group or whether it's on a national or international stage, go on podcasts, write on a blog. If you don't have one, create one or write on Medium or whatever, but share what you know, share what you've learned and try and disseminate that information as best as possible.

Lea Alcantara: Oh, thanks, Aaron. I feel like we've gotten a flood of information because this is a big rabbit hole we could just continue talking about because it really affects every process of developing a site.

Emily Lewis: Yeah, it's basically like let's talk about web design. [Laughs]

Lea Alcantara: Yeah. [Laughs]

Aaron Gustafson: [Laughs]

Lea Alcantara: Yeah, essentially.

Aaron Gustafson: Six hours later. [Laughs]

Lea Alcantara: [Laughs]

Emily Lewis: [Laughs]

Lea Alcantara: But 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, Aaron?

Aaron Gustafson: Sure.

Lea Alcantara: Okay, first question, morning person or night owl?

Aaron Gustafson: Morning person.

Emily Lewis: What's one of your guilty pleasures?

Aaron Gustafson: Ooh, that's a good question. I go from musically, Bon Jovi.

Emily Lewis: [Laughs]

Lea Alcantara: [Laughs]

Aaron Gustafson: Or White Snake, either one.

Emily Lewis: [Laughs]

Lea Alcantara: Oh my God. What software could you not live without?

Aaron Gustafson: The Terminal.

***Timestamp**: 01:19:57*

Emily Lewis: What profession other than your own would you like to try?

Aaron Gustafson: When I was a kid, I wanted to be a marine biologist, and I have a hobby of maintaining a reef tank, so I probably like to get back into that and have that be my next career. I don't know.

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

Aaron Gustafson: I think I would be really crappy at HR.

Emily Lewis: [Laughs]

Lea Alcantara: [Laughs]

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

Aaron Gustafson: Hmm, that's a good question. Probably, Sluggo's North, which is a great little punk rock vegan restaurant.

Emily Lewis: Are you a vegan?

Aaron Gustafson: I am what I refer to as a non-sanctimonious vegan. [Laughs]

Emily Lewis: [Laughs]

Aaron Gustafson: And not terribly strict vegan. I still do honey and I'll still do occasional animal products, but mainly I try and keep it at 5% or below, so mostly vegan, but not strict.

Lea Alcantara: If you can meet someone famous, living or dead, who would it be?

Aaron Gustafson: I really wished I got an opportunity to meet Johnny Cash.

Emily Lewis: Yeah. If you could have a super power, what would it be?

Aaron Gustafson: Oh, I always loved Nightcrawler's line of sight teleportation. I thought that would be awesome.

Emily Lewis: [Agrees]

Lea Alcantara: Very cool.

Aaron Gustafson: So I'm lazy, I don't need to walk places.

Lea Alcantara: [Laughs]

Emily Lewis: [Laughs]

Lea Alcantara: What is your favorite band or musician?

Aaron Gustafson: Gosh, I like so many different kinds of music. Lately, I've been getting really into Lil Simz who is a British rapper.

Emily Lewis: [Agrees]

Aaron Gustafson: She started putting records together I think when she was like 13 on Bandcamp and she's amazing. So I would say her right now.

Emily Lewis: Last question, pancakes or waffles?

Aaron Gustafson: Oh, that's a really tough one.

Emily Lewis: [Laughs]

Lea Alcantara: [Laughs]

Aaron Gustafson: I really enjoy a good pancake with like a crispiness to it.

Emily Lewis: [Agrees]

Aaron Gustafson: But if they're not crispy, if they're just kind of soft…

Emily Lewis: Limp.

Aaron Gustafson: Yeah, then I'm not as big a fan, but like a really good crispy pecan pancake is like hard to beat.

Lea Alcantara: [Agrees]

Aaron Gustafson: But if it's not crispy, then absolutely waffles, but again, it has to be crispy. If it's not got the crunch, then I don't want it.

Lea Alcantara: [Laughs]

Emily Lewis: [Laughs]

Lea Alcantara: Awesome. Thanks, Aaron. That's all the time we have for today.

Aaron Gustafson: Thank you very much,.

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

[[Music starts]

Aaron Gustafson: So they can find me on Twitter as @aarongustafson. I also have aaron-gustafson.com.

Emily Lewis: Awesome. This was a great discussion. Thanks for coming back to the show.

Aaron Gustafson: Thanks for having me.

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 sponsor! Thank you, Craft Commerce.

Emily Lewis: We'd also like to thank our partners: Arcustech and Devot:ee.

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! And if you really liked this episode, consider donating to the show. Links are in our show notes and on our site.

Emily Lewis: Don't forget to tune in to our next episode when Brenda Storer joins us for a primer on SVGs. 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: 01:23:15
