EE Podcast #63 Progressive Enhancement w/ EE

[Music]

Lea Alcantara: You are listening to the ExpressionEngine Podcast Episode #63, Progressive Enhancement with EE, which we will discuss with our special guest, Aaron Gustafson. I'm your host, Lea Alcantara, and I'm joined by my fab co-host, Emily Lewis. This episode is sponsored by mithra62. CartThrob 2 is officially out of beta. To celebrate the launch, CartThrob is making a discounted package available that pairs CartThrob 2 with popular CartThrob 2 add-on, CT Admin, at an introductory price of a $189 for both add-ons. Save $20 for a limited time. Available on CartThrob.com

Emily Lewis: The ExpressionEngine Podcast would also like to thank Pixel & Tonic for being our major sponsor of the year. [Music ends] Hi Lea, how are you doing?

Lea Alcantara: Pretty good. It's Reading Week here for universities in Canada, so I have a bit of a teaching break this week so that gives me a little bit of a breather. How about you?

Emily Lewis: Oh, I'm not too bad. I got hit with a cold last week and I'm still recovering but definitely feeling much better, and I'm really glad to hear you are finally getting a break from your insane schedule.

Lea Alcantara: [Laughs] It's a little nice, yeah.

Emily Lewis: [Laughs] Before we introduce our guest for today, I'm curious, what is your experience with the progressive enhancement?

Lea Alcantara: To me progressive enhancement always meant creating websites that were semantic and had browsers and accessibility in mind, so pretty much proper web standards practices. So to me progressive enhancement is a reminder that technology and audience is vast and to accommodate for that while still pushing the web forward. What's your take?

Emily Lewis: To be honest, I've always been confused about progressive enhancement versus graceful degradation.

Lea Alcantara: [Agrees]

Emily Lewis: But in preparation for our podcast today I read Aaron's adaptive web design, which clarified things for me, and I'm really excited that I've pretty been following progressive enhancement techniques for quite a while now, which makes it a really segueway to introduce our guest, Aaron Gustafson who authored *Adaptive Web Design*. Aaron has over 15 years experience in the field specializing in the front end. With his wife Kelly, Aaron runs Easy! Designs, a boutique web consultancy in Chattanooga, Tennessee, and when he's not busy running his own business, Aaron is the group manager of the Web Standards Project and an invited expert to W3C's Open Web Education Alliance. He also serves as technical editor for a list of A List Apart, and is a contributing writer for .net Magazine. Welcome, Aaron.

Aaron Gustafson: Hey, thanks for having me guys.

Emily Lewis: We are excited to have you on. Both Lea and I spent the past few days reading your book, and I have to say I quite enjoyed it.

Aaron Gustafson: Awesome. Hopefully it didn't take you too long to read the whole book.

Emily Lewis: Well, no. That was what I was about to say. I'm loving this trend I see in the industry with these much shorter tech books. You can sit down with a cup of coffee on the morning on Sunday and kind of get through it, and I love having the digital format so I can make notes and highlights right in the device.

Aaron Gustafson: Very cool.

Lea Alcantara: Yeah, and I also liked that. Honestly, I'm almost old school that I prefer the printed version because even though that it's a shortened sweet book, I feel like I can absorb more of the information sitting down with the book, physically feeling the paper and like pausing and making notes with a pen and paper. I sound so old school for someone who works on the web, but I actually quite prefer that when I'm trying to learn something.

Aaron Gustafson: So actually that kind of brings up an interesting point about this because, I mean, when I initially was thinking about doing the book, obviously digital was a big portion of what I was focused on and interestingly enough progressive enhancement works for EPUB as well, which we could talk about if you are interested. But one of the things that we did because we decided to self-publish the book and we actually instead of just doing it as another Easy! Designs' product, we decided to actually create a publishing company so we could eventually do other people's books, and one of the things that we did was we actually took a lot of time to look at paper and to look at dimensions and to actually have the printer that we are working with take the papers that we thought we wanted to work with in terms of cover and paper stock and actually bind it together for us to give us a sense of what the weight was going to be of the book and how it was going to feel in our hands, what the tactile experience was going to be. Because to me, I've got a huge library of tech books.

Lea Alcantara: [Agrees]

Aaron Gustafson: Just shelves and shelves and shelves of books and a lot of them, I don't know, it seems like the medium doesn't necessarily fit the message so you will end up with these really substantial intellectual volumes from pragmatic publishers, for instance, or somebody, O'Reilly Publications, but they are printed on such fine newsprint almost that the book weighs nothing so it feels inconsequential. So we really kind of took a long time to think about what it was that we are putting into the physical version of the book. So I guess my question to you, Lea, is how did you feel the reading experience was in terms of the physicality of the book?

Lea Alcantara: Well, I have to say I loved it, especially because I actually come from the print design background way, way back when I started my design career so I appreciate the details of the paper, the weight of the cover stock and again there is just something about paper being easy to read. I know there is e-paper out there and there are different ways you can have a nicer reading experience with e-books, but all of them are trying to mimic paper.

Aaron Gustafson: Right.

Lea Alcantara: So there is nothing like the original, in my opinion.

Aaron Gustafson: It's true. I mean, there are some really neat things that you can do in the e-book arena such as doing audio and video like being able to embed video samples and stuff. I think it's really helpful for helping people to understand.

Emily Lewis: I love that.

Aaron Gustafson: Yeah, for helping people understand things like accordion interfaces and tab interfaces and stuff.

Lea Alcantara: Sure.

Aaron Gustafson: I think video is really awesome for doing that. When you read it, Emily, did you happen to use Readmill or did you read it in iBooks, or what was your…

Emily Lewis: I have a NOOK Color.

Aaron Gustafson: Okay.

Emily Lewis: So it was in that, and I was able to play the embedded media, which I thought was really cool. That was the first publication I had that had that embedded in it, but I completely relate to Lea's comments and your own investment in the materials in the print books because in the past, print has always been my preference. I too felt I learned best by making notes by hand. I've just slowly been trying to move away from that because I've moved enough times that I can't continue moving these books each time.

Aaron Gustafson: [Laughs] I completely understand you as somebody who have about 3,000 CDs before we moved one time.

Lea Alcantara: Wow!

Emily Lewis: [Laughs]

Aaron Gustafson: I can understand the cost and the physical toll of having to move massive amounts of books and just media in general. I mean, one of the things that I guess is really interesting about the electronic thing apart from the video is like I actually was part of kind of a pilot program by Readmill to do commentary on the book, sort of like that director's commentary.

Lea Alcantara: [Agrees]

Aaron Gustafson: So I did one of my many reads through the book on the iPad in Readmill and highlighted sections and made kind of notes about my notes and notes about various things that I was talking about in the book, and a few jokes on the side on why things ended up being, and that was kind of an interesting experience to me because I've always been a film nerd as well.

Lea Alcantara: [Agrees]

Emily Lewis: [Agrees]

Aaron Gustafson: And so when DVDs came out, all of a sudden you had these alternate audio tracks and sometimes people scripting at the bottom, and like the Ghostbusters DVD, it's awesome because it's got like the MST3K style. People scripting at the bottom doing audio commentary. I think one of the Muppet movies did that as well, and being able to kind of bring that to the book world, I mean, both of you have read the book and you've probably read my other stuff as well, so I'm very full of tangents so there are lots and lots of footnotes in the book because I tend to think tangentially, but it was kind of interesting because in some cases I was doing notes on my footnotes and just kind of take notes here in the sides and stuff.

Lea Alcantara: [Laughs]

Emily Lewis: [Laughs]

Aaron Gustafson: And it was kind of neat to have the freedom to that, so I was pretty excited about that opportunity when they came to me and were asking me to do that for the books. [Laughs] I recommend reading it in Readmill, it's fun.

Emily Lewis: I'll have to check that out, and I've noticed that a lot of the comments I've seen on Twitter and such, everyone is really enjoying the book. So let's dive into the topic. The book is called Adaptive Web Design, but you speak about progressive enhancement. What's the difference between adaptive web design and progressive enhancement, or are they sort of one and the same in your view?

Aaron Gustafson: So to me they are kind of one and the same. I mean, the title *Adaptive Web Design*, it wasn't really an attempt to like draw a line in the sand and say, "This is what we should call what we do." It was very much an attempt to differentiate the book from a marketing standpoint from the one other progressive enhancement book that's out there which is *Designing With Progressive Enhancement* by the folks at Filament Group, which is a great book.

Emily Lewis: [Agrees]

Aaron Gustafson: And it's actually one of the books that I recommend in the kind of further reading section in the back of the book. I wanted to differentiate it from that and make sure that there was no ambiguity and frankly as a topic with progressive enhancement, it's not all that sexy of a term, right?

Lea Alcantara: [Laughs]

Emily Lewis: [Laughs]

Aaron Gustafson: And really we worked through a bunch of different title options trying to think of what it was that really the book was speaking to, what was the promise that the title needed to kind of get across in terms of the nutshell of what the book is about. *Adaptive Web Design* was kind of a way to summarize that and then the subtitle, which is *Crafting Rich Experiences with Progressive Enhancements* kind of broadens on that a little more. To me, *Adaptive Web Design* is very much another way of saying progressive enhancement. It's kind of a summation of all the different techniques that we use.

So to me, adaptive web design and progressive enhancements encompasses things like responsive web design, which is more focused on the visual aspect of a website, so fluid grids, fluid images, those sorts of things, media queries, it takes it a little bit further because adaptive web design and progressive enhancement is about the complete experience of thinking about it from the point of pros authoring the actual copy of the website through the semantics that are being applied through the CSS that is being applied and then through your JavaScript and eventually your accessibility enhancements like ARIA and stuff like that that people tend not to think of or tend to think of as very specialized aspects of web design and really showing how it's a continuum and how we can take advantage of the ways that these different technologies are created and have been drafted to build an ever improving experience for users.

So really kind of crafting that awesome experience and making sure that there are technological restrictions to users that no matter what device somebody is on whether it's an old BlackBerry or the latest iPad or something like that, that they are going to be able to access your content and be able to complete the necessary tasks that they come there to complete or to get the information that they are looking to get from your service.

Emily Lewis: As I mentioned earlier on, I always had a bit of confusion between progressive enhancement and graceful degradation. Your book does a good job of sort of clarifying that. Could you do so for our listeners?

Aaron Gustafson: Sure. The way that I kind of looked at it is that graceful degradation for a long time was the de facto standard. It was you built the latest and greatest browsers and then you did some modicum of testing for older browsers just to make sure that they weren't having horrible errors, but you basically assumed that they were going to have a bad experience. Progressive enhancement, when that idea was first created by Steven Champeon back and it was in 2003 at SXSW was the first place that he started talking about it. But he kind of turned that idea on its head and he said we needed to work from the content out, which at that point, that framing of it makes it seem like progressive enhancement and graceful degradation are at odds, which in some ways they are but I think it's more of an attitude that's at odds between the graceful degradation camp and the progressive enhancement camp. But in reality, the two are fairly closely linked because everything that you build using progressive enhancement, everything that you do to build layer upon layer

to create an improved experience and a tailored experience for users of particular devices, that automatically is going to degrade gracefully. So everything that you do with progressive enhancement is taking advantage of graceful degradation or is employing graceful degradation.

But not everything that's built following the graceful degradation philosophy is progressive enhancement. So progressive enhancement is kind of a specialized form of graceful degradation where you are actually giving a little bit more consideration to the user and realizing that we don't know a lot of times what the user is on or how important the user is that is on a particular device. So we can look at our statistics and say, "You know, old BlackBerrys, okay, that's 0.001%." But you don't know if there is a businessman or woman out there who is using that aging BlackBerry that has billions of dollars that they want to invest in your product because the analytic stuff we have does not give us that sort of introspection into our users, although Facebook would probably like to get there at some point.

Lea Alcantara: [Laughs]

Emily Lewis: [Laughs]

Aaron Gustafson: But we don't know and we have to come to grips with the fact that we don't know and we cannot be 100% assured of everything so why not try and give everybody a great experience. Yes, we can work with the fact that some people are not going to have as high-fidelity of an experience as other people.

Emily Lewis: [Agrees]

Aaron Gustafson: But as long as they are having a positive experience and can do what it is that they've come to your website to do, then all is well for me.

Lea Alcantara: I think that's a good point that you make. I think some people get hung up over the fact that progressive enhancement or graceful degradation is trying to give everyone an equal experience because that is just not possible.

Aaron Gustafson: Yeah.

Lea Alcantara: But you can give everyone a good experience, it's just that the level of how great that experience is, is different depending on what device they are using, et cetera, and so forth.

Aaron Gustafson: Yeah, there were actually two really interesting pieces that I read in the last, I guess, it's a couple of weeks with two individuals talking about how progressive enhancement and graceful degradation are really two political philosophies, which I found very fascinating because they were both obviously Poly Sci majors and that was not area, but I'm a political junkie.

Emily Lewis: [Agrees]

Aaron Gustafson: I mean, when I tend to look at graceful degradation and progressive enhancement, it seems like the people who fall into the graceful degradation or as they are more likely to self-identify now, the hardboiled design folks, they tend to view progressive enhancement as almost a communistic approach to web design where you are trying to just take from the rich to give to the poor and make sure everybody has the exact same experience sort of thing, whereas to me, progressive enhancement is much more of an egalitarian approach in that we don't know what somebody's experience is, we don't know what somebody's upbringing is. Not everybody has the same advantages, but we still want to make sure that everyone is able to live a productive life or do what it is that they are trying to do, which I think is kind of a better way to look at it as opposed to trying to make the playing field level for everyone.

Emily Lewis: Yeah.

Lea Alcantara: [Agrees]

Emily Lewis: It reminds me of the analogy you have on the book of the peanut M&M.

Aaron Gustafson: Right.

Emily Lewis: Like everyone, the peanut is the center, everyone gets that goodness, and then you add the layer of chocolate for the people who have the device that supports chocolate. [Laughs]

Aaron Gustafson: [Laughs]

Emily Lewis: And then you add the candy coating for the people who have the ability to have that, but everyone gets the core.

Aaron Gustafson: Yeah. Everyone gets a perfectly valid sap. Well, of course, except for people who have peanut allergies.

Emily Lewis: [Laughs] Oh, right.

Aaron Gustafson: But each of those steps along the way for the American audiences, the peanut, the goober and the peanut M&M, and there are other brands of chocolate-covered peanuts as well out there I'm sure in other countries, each of those is certainly a valid snack and is a good snack experience and just with some, you could argue with the better snack experiences. Although some people might really like the peanut.

Emily Lewis: [Agrees]

Aaron Gustafson: So there are some people who do. I mean, I don't know if you guys saw the Guardian article, the guy was talking about how graphic designers are ruining the web.

Lea Alcantara: Oh, yeah.

Emily Lewis: [Agrees]

Aaron Gustafson: Which was kind of funny. Apparently, he really likes the peanut. He really wants that just very basic experience with no visual enhancement at all, so I mean, maybe there is a group of people who like that. I know when A List Apart initially started removing styles from Netscape for back in the very, very early days when I first joined there, they actually saw an uptick in the number of people who were using Netscape browsers because I guess it created a more streamlined experience. I'm not quite sure what the reasoning was and I'm not sure if ever found out what the reasoning was, but they actually saw their percentage of usage go up in the browsers that they were no longer delivering styles to which they were just taking advantage of CSS parsing errors to deliver CSS to modern browsers but not to Netscape.

Emily Lewis: In terms of building these layers, can you, on a sort of superficial level, talk about these layers or these building blocks of progressive enhancement?

Aaron Gustafson: Sure. Well, the baseline is definitely the core of your experience and the web is essentially text in HTTP. That's the absolute lowest level of experience that anyone who has ever fired up links as a text-based browser will understand that, and that's really your first level of interaction with any website is. If it's got really clear, well-written copy, that's a usability enhancement right there. That's something that's going to help you sell products or make sure people understand what it is that they are signing up for, if they are signing up for your product, and then HTTP is obviously how we move from page to page within the web, but also how we are able to submit data via forms and stuff like, how we are

able to collect information or allow people to publish information as in the case of content management systems. So that's kind of the absolute baseline, the zero layer of progressive enhancement, and then upon that we apply semantics which are done via HTML, so you have the semantics at different layers of HTML, everything from HTML 1 all the way up through HTML5 and the neat thing about the way HTML and the way that CSS are kind of drafted or spec'ed out to operate within a browser is that browsers ignore what they don't understand. So that's what allowed us to use an HTML5 form control type such as date or email or tel or something like that, and IE 6 will just ignore that and make it a standard text field. So we can take advantage of these modern approaches to HTML and not have to worry about it in older browsers.

Semantics is kind of the enrichment of the content by providing more meaning behind the words and then we start applying the styles, and my approach to applying styles is actually to approach it in terms of facets which is something that I talk about in the book, beginning with your typographic styles and then moving into your layout styles and then finally your color styles, and I do that to kind of compress the style sheets a little bit because it allows you to take advantage of the shorthand within the layout styles for borders and the like and then override the border color in your color styles.

But that facet approach works really well because it allows you to expand what it is that you are targeting with CSS. So if you've got your typography styles, obviously that's something that every browser can understand and then as you begin assigning your layout styles, you can do your base layout styles as kind of your mobile first layouts, so your single column view, your very simple margins and padding so that you are not taking up too much room, and then you can use media queries to adjust the layout, moving up in the browser size based on how much real estate you have available to you in creating those different interfaces and creating different layouts and such and then finally tapping on or tucking on the color styles and then maybe effects on top of that, but it gives you a clean separation which makes it easier to maintain your style sheets and to manage growth of a website in addition of new features and such over time.

And that's kind of a direction I've gone with my style sheets over the last, I would say, five or six years and it has worked really well for us in all the projects that we've done and it's actually nice how it prepped us to be ready to work in a mobile first way and to be ready for media queries when we finally got media queries, and those style sheets could exist as one big style sheets with lots of media queries within it or it could be separate style sheets which might be better if you are targeting specific older devices, older handheld devices because you can use your media queries in the media attribute of the linked style sheet, which would stop any phone that doesn't understand media queries or that won't assign those styles because if it won't implement that media query, it will ignore those styles and not even download that external style sheet. So you can play it around a lot with that to kind of tune the performance of your site and optimize the performance of the site.

So that's kind of the CSS layer and you can apply your styles using those facets, and then there is the JavaScript layer which to me the best approach to JavaScript is really kind of ala carte approach where you are creating individual interactions and making decision about whether to apply that particular interaction, let's say, converting content into a tap interface, for instance, you make the decision as to whether or not to make it a tab interface based on whether the necessary JavaScript components are there to do it, and I definitely advocate for having JavaScript create the extra markup or inject the extra styles that those individual components, those interfaces need because normally if it doesn't have JavaScript support, they are not going to be able to run that widget so why force that user to download the extra CSS or the extra HTML that would only be required for use in that interface scenario.

Then the final layer, and this is kind of debatable as to whether you want to call it a final layer, but the layer of ARIA as accessibility is really HTML semantics, mostly using the role attribute, but then there is a lot of manipulation of that that takes place within JavaScript because ARIA is really useful in complex interfaces like tab interfaces and accordions and the like. So I kind of separate it out as kind of the final top layer, but it's really kind of spread throughout the process.

Emily Lewis: Yeah, I'm glad you clarified that. That was actually one of my questions because of how you had laid it out in the book, I was starting to question. I usually write my ARIA when I'm writing my semantic markup.

Aaron Gustafson: [Agrees]

Emily Lewis: And I was thinking, "Gosh, should I be holding off until the end because that's the last layer?" So it's really just a way of envisioning the layers, it's not really the process.

Aaron Gustafson: Yeah, and I think the way that I was approaching the book is that there are a lot of people, most of the people reading the book have a fairly good understanding of HTML, at least HTML 4.01, right? Maybe not HTML5 yet, and probably have a decent sense of CSS, and maybe terrified of JavaScript, but kind of having ARIA as this top thing. It's something that they are probably aware of, but maybe not all that cognizant of how to use it properly and what the different implications of decisions that you make in terms of scripting or in terms of style and such, and what impact those have on accessibility. So to me, it kind of made sense to not muddy the waters too much in those other chapters and to really address accessibility kind of on its own, even though it really is spread throughout all those other sections.

Emily Lewis: [Agrees] You had mentioned in your discussion about sort of embracing how older browsers will sort of ignore something that they don't support. I think you called it fault tolerance.

Aaron Gustafson: Yeah, it's fault tolerance or in the case of CSS, it's specifically called parsing errors, browser experiences a parsing error and then decide based on the rules of parsing errors in CSS, it does one of the several things in terms of ignoring individual rules and ignoring entire rule sets or entire style sheets, et cetera.

Emily Lewis: Now, CSS and HTML follow that, but what about JavaScript?

Aaron Gustafson: JavaScript is the tricky one, mainly because it's a programming language. So the browser needs to understand everything that you are using in order to be able to not throw an error, and if it doesn't understand something, it has to halt execution of the program that you've written and that's what causes JavaScript errors. So JavaScript can't be fault tolerant for that reason. There have been a couple of different ways that browsers have tried to address allowing you to move forward in terms of what's supported in JavaScript. For a while, you could state a JavaScript version for Firefox as part of the MIME type that you are saying, so text/javascript, and I think you would use like a semicolon, and it was like version 1.7 or something like that. I'm forgetting the exact syntax, but it was something to that effect that would allow you to take advantage of newer constructs in JavaScript and it would interpret the JavaScript using that newer interpretation engine, so you could use things like let statements which is kind of like having scoped variables, so you could say, "Let a particular variable equal this within the context of this loop that I'm in with that variable would not exist outside of the loop." But because let was not a keyword prior to whichever version of JavaScript it was that that was drafted, and I think it was 1.6 but it may have been 1.5, any other browser would not know what to do with the let keyword. So that would cause an error in those browsers. So JavaScript, because it is a programming language or scripting language, it can't follow that kind of same fault tolerant nature that the CSS and JavaScript do which is why we have to do tests to see whether particular methods that we want to use are available or particular properties are available, and libraries have certainly helped us a lot with that because they've kind of abstract the way a lot of the browser issues in terms of event models and ways of accessing content and manipulating them and such.

So if you are using jQuery, you don't have to worry about it as much, although if you are using jQuery and you are also doing a lot of stuff for mobile, not every mobile browser actually supports jQuery across the board, and I believe jQuery is getting ready to drop support for IE 6 if they haven't already. So there are some considerations as far as that stuff goes moving forward, but traditionally that's where JavaScript Library has helped out a lot is by kind of abstracting that stuff and doing the tests to make sure that things like an element by its last name and stuff like that are available to you.

Lea Alcantara: So actually, I was reading through Twitter some conversations about JavaScript being enabled or not enabled and you made some comments over trying to make sure that people that have devices that don't deal with jQuery still get the proper content, so my question is then, do you just deal with functionality that you know for sure will be done without JavaScript when you are dealing with multiple platforms and making sure everyone has got the same experience? How do you deal with that?

Aaron Gustafson: I mean, it really comes down to kind of designing what the progressive enhancement experience is going to be for a particular age or a particular widget.

Lea Alcantara: [Agrees]

Aaron Gustafson: So in the case of the tab interface site, I think this is a pretty good example that I walk people through a lot, a tab interface requires JavaScript in that being able to move from section to section within a particular content block with little tabs sticking out.

Lea Alcantara: Yeah.

Aaron Gustafson: The way that I approach a widget when I was actually trying to make it a really super accessible, very progressive enhanced version of a tab interface is that I started out by looking at the markup and I realized that essentially if you think about document outline, which a lot of people don't, but if you think about document outlines, they are naturally created by heading levels.

Lea Alcantara: Yes.

Aaron Gustafson: And instead of actually breaking up content by actively putting a div between certain sections of content, if I took a container element and had it classified as tab, for instance, if you go through that and look for the first heading level that it encounters, and if that's an h2, I can say, "Okay, well, based on the fact that an h2 is the first heading level that's encountered, I know that any other h2s are going to denote sections that are at the same level as that in the document outline." So I can write my markup and just have that container element and then within it have my headings, my paragraphs, my list, my definition list, whatever the content is for each of the sections and not have any other additional markup, and then using JavaScript, I can come through and say, "Okay, here is something classified as tab that I want to turn into a tab interface." I can run all of my checks to make sure that the various methods that I want to use in order to create this like document.getElementById, for instance, or getElementsByClassName, or getElementsByTagName, or something like that, which all browsers pretty much support across the board at this point, but back in the day was not necessarily a given.

But I can go through and say, "Okay, here is a heading level." And then I can split that content based on that h2 throughout and create the markup that I needed in order to generate the tab list at the top as well as the containing blocks for each of the sections of that tab interface and apply all of the different events that I need to handle in terms of when you click on a tab, hiding one tab panel and showing another tab panel, as well as adding the keyboard controls and adding the ARIA roles and other ARIA properties and such in order to make the interface fully accessible, and what you then end

up with is that for somebody who comes to that interface with no JavaScript, they just get the content nicely organized with heading levels and all of that sort of stuff and then somebody who comes to the page that has a more hi-fi browser, they can get that enhanced experience in a browser that even doesn't support ARIA, for instance, can have that enhanced tab interface experience and then one that has ARIA support on top of that can provide a more accessible tab interface experience.

Then the interface itself, the JavaScript when it runs in order to build that interface also injects or triggers the application of styles to build that interface. So all the styles are maintained separately from the JavaScript and the JavaScript simply flips a switch, and the example of the tab interface that I usually use, I switch it from having a class of *tab* on the containing element to *tab-on* and all of the styles for the tab interface are treed off of that class of *tab-on*, so you don't have styles being applied to build the tab interface that won't actually operate because the JavaScript that would make it operate isn't available because there is a JavaScript error or something. So the whole creation of the interface in the application of styles to the interface is really governed by the code that actually runs the interface as well, and that's kind of what I mean by having your interfaces, your JavaScript built in an ala carte fashion where they are maintaining control over kind of their little portion of the experience but aren't relying on anything else or aren't depending on things to be managed elsewhere or to trigger them from elsewhere, which makes it easier to kind of drop that script in to a new site and have it just work as opposed to having to reconfigure it each time.

Emily Lewis: Thus far most of what you've talked about is all front-end.

Aaron Gustafson: [Agrees]

Emily Lewis: I'm curious, I know you work with ExpressionEngine quite a bit. Is there anything with regard to EE that is either a challenge or a benefit to progressive enhancement techniques?

Aaron Gustafson: I mean, we do a lot of sites in ExpressionEngine and it has worked out really well for us in terms of being able to kind of roll with our progressive enhancement needs. I like to maintain my style sheets separately and I do want to do server side concatenation of those so we do that via ExpressionEngine and then cache out the final files so that we are not having to recombine the CSS and we maintain those each as templates just from a managerial standpoint. That seems to be the easiest route for us, but we also use a lot of the templates as sort of partials for JavaScript to make an AJAX called the pull-in content that it might want or to generate piece of JSONP and such, and ExpressionEngine has been really good about that as well.

One of the new things that we did when we re-launched our blog was to actually take our comment, or not our comment page but our main article page for each of the blog posts that we do and in that one template, we account for whether or not there is a segment of the URL that has comments in it, and I guess it's the fourth segment, so it's usually the name of the article and then after that we have */comment*, and we have the template handling it in such a way that if comment is present it goes ahead and embeds the comment feed into the page. But if comment is not there, it instead injects a link into the page that says, "Follow this link to view the comments and add your own."

Then we are using what's called the AJAX include pattern where we are using jQuery to go through and actually pull in the comment portion of the page via AJAX after the page has finished loading and inject that into the normal page, replacing the link. What that gets us is it gets users the content that they are coming to read so that the actual page itself immediately and then loads in the comment after fact via AJAX if JavaScript is available. But if JavaScript is not available,

we include a link that takes you to the same page again but with the comment thread embedded in it and it's an anchor reference down to that.

So that for us kind of fills all of our needs in terms of speeding up access to content for the 99% of users that are just coming to read as opposed to comment and contribute back, but it also covers our bases in terms of JavaScript support and JavaScript support for being able to comment or to read existing comments, and the ExpressionEngine worked really for that because we were able to essentially abstract out the comments portion and pull that into the response to the AJAX query to drop in the comments as well as dropping it into the comment version of the template when that fragment is available, or when that segment is available. So that first step has been really handy in terms of ExpressionEngine.

Lea Alcantara: You mentioned using include templates, and I know I was looking through your EECI slides and you are using a lot of those as examples.

Aaron Gustafson: [Agrees]

Lea Alcantara: Any reason why you prefer using templates for your include as opposed to something like snippets or Low Variables or something like that?

Aaron Gustafson: I've done Low Variables before and we've just started using EE 2 more and more. The site that I was showing examples from is actually still 1.6, so snippets didn't exist in it.

Lea Alcantara: Okay, yeah.

Aaron Gustafson: So that was the reason that all those examples don't show snippets, but snippets is something that we are very interested in doing more of and as we are converting a lot of our 1.6 sites over to 2, we are shifting into using snippets as opposed to using embeds.

Lea Alcantara: [Agrees]

Emily Lewis: I was also looking at your slides from EECI, and I noticed that you also referenced just a handful of add-ons that you've used that sort of helped a bit with your semantic markup.

Aaron Gustafson: Yeah. Let's see, let me find my slides here.

Emily Lewis: [Laughs]

Aaron Gustafson: You probably have them in front of you.

Emily Lewis: Yeah, I'm looking at them right now. I'm kind of pleased. I didn't even know there was an add-on for microformats for sort of making it easier to generate that.

Aaron Gustafson: [Agrees]

Emily Lewis: That's something I've always thought about because it can be a bit of a pain to hand code all of the classes and attributes that are necessary to properly mark something up.

Aaron Gustafson: Absolutely, yeah. The CI microformat one is one that exists for CodeIgniter, but obviously has been applicable to EE 2.

Emily Lewis: I'm wondering in terms of working with ExpressionEngine and doing this progressive enhancement, is there anything with regard to EE control panel itself that sort of fails or doesn't fail with regard to progressive enhancement? Have you ever had a client had to work with the control panel and had a low-fi browser or low-fi experience?

Aaron Gustafson: We haven't run into that before. It's something that I am interested in kind of figuring out whether it's really necessary. To me, when you start operating in the realm of like the control panel itself or in the realm of an application-linked base camp or even Gmail to an extent, when you are operating in an area where you essentially required somebody to sign up for a service, I think that you can have certain requirements just to keep yourself from going insane.

Lea Alcantara: [Agrees]

Aaron Gustafson: When 37 signals decided to drop IE6 support, a couple of years back, I didn't really think that was all that bad of a thing for them because it makes sense from their standpoint because they are a Software as a Service, and they have to spend their maintaining their site and such. With that said, I don't think that they are reliance on JavaScript is necessarily the best route to go because things happen.

Emily Lewis: [Agrees]

Aaron Gustafson: People are always saying, "Well, people have to consciously turn the JavaScript off. So if they don't have JavaScript, that's their problem." Well, that's probably 80% of the people maybe turning the JavaScript off, but there are firewalls that block JavaScript. If there is a JavaScript error in the page, it can cause all of the JavaScript execution to stop. I mean, we saw that just recently with the Gawker Media sites where literally every site on their platform was down because of a JavaScript error.

Lea Alcantara: [Agrees]

Aaron Gustafson: And then there are people who are like, "Well, you know, my JavaScript is awesome, and I will never have an error in my JavaScript."

Emily Lewis: [Laughs]

Aaron Gustafson: But if you rely on any third parties that get JavaScript into the page, I mean, I've gotten that version of jQuery from the Google AJAX library that have caused stuff not to work.

Lea Alcantara: Yes.

Aaron Gustafson: So if you are relying on anybody but yourself or you question whether you might accidentally type an S where you shouldn't have when you were hitting save on a JavaScript file, it's best to be able to have a site that would be able to continue to operate regardless. So I mean, I think it makes sense to not 100% rely on JavaScript within things like a control panel like EE's control panel, but I think that to some extent it's also okay to have certain requirements for accessing the control panel in terms of browsers that you are willing to support and to make JavaScript a requirement, but maybe to make sure that things can still operate okay if something happens with a third party plugin that all of a sudden breaks the site or breaks the control panel because of it injects some bad JavaScript or something.

There are so many things that are beyond our control even though we are try and control as much as we can, but I'd rather have stuff that's going to be able to stand up and continue to work rather than have something that's just going to fall apart.

Emily Lewis: And I'm curious, you didn't mention it too much in the book. How do you test? What's your approach to it?

Aaron Gustafson: So a lot of our testing, we will test with JavaScript on and JavaScript off. We do most of our development in browsers like Chrome and Firefox. We have several virtual machines that we use for testing, with various versions of Windows. We do have a Windows machine, too, but if we can avoid firing that up, we will. But we've got a couple of XP installs. I had been using VMware for a long time for my virtual machines. I've been slowly migrating over to using VirtualBox because it's free. I think my setup right now is I've got an XP install that has IE8. As the installed version, I'm using IE Tester as a piece of software inside of the virtual machine to test the other versions of IE. I did have the standalone installs of IE6, 7, and 8 that were for .Net before Microsoft asked them to kindly take them down. So we've had them on virtual machines as well. I think I've got that on my Windows 7 VM. On Windows 7 I've got obviously a couple of Firefoxes and Chromes, Safari, Chromium and then I've got IE9 and IE10 running on that and then I've got a Windows 8 VM that I'm doing some beta testing of IE10 stuff just to make sure things aren't falling apart too badly, and then lots of other browsers on my Mac. I think I've posted a photo to Twitter a couple of days ago that showed my browser archive folder and I think I probably got about, I don't know, 12 versions of Chrome and stuff like that that I've kind of collected along the way.

Emily Lewis: [Laughs]

Aaron Gustafson: Somehow when we were working on a Chrome app that we did about a year and a half or two years ago, we had to switch over to the Dev channel for Chrome and somehow in the process of doing that, it botched my Chrome install to the point where it will automatically update which has been great because it allows me to have the old version of Chrome and download new version of Chrome and have them each to be standalone.

Emily Lewis: Oh.

Aaron Gustafson: So that's actually been kind of a blessing. So each time a new version of Chrome comes out, I obviously have to go and download it, but it's nice because I can actually test to see whether things are regression bugs, for instance. So Chrome, because of their frequent update, they sometimes introduce bugs that didn't exist in prior versions, like I've been revisiting this Chrome app that we did to actually make it work for Firefox and for Opera and for Internet Explorer, and we noticed in the latest version of Chrome and Chrome 17, it drops out background images at weird times. It wasn't doing it in Chrome 16 or before and when we put this app out, I think it was originally like Chrome 9 or something. It's something that they've introduced as a new bug and hopefully will be addressed, but it still seems to be in Chrome 19 which is the current beta release or dev release, I'm not sure which channel that was on, but anyway, so it's helpful to be able to have those different versions because then we can actually look at a site if a client comes to us and says, "Hey, this isn't working now." We can say, "Okay, this is something that we will fix or this is something that is a problem that the browser just caused and they will hopefully pick it up in their next release that will be just transparently behind the scenes for their users if it's Chrome."

Lea Alcantara: So mentioned all the browsers and items, but I don't think I caught when you actually start looking at the different browsers.

Aaron Gustafson: Oh, so…

Lea Alcantara: Like do you finish the site in the main browser like the latest browser that you are targeting and then look at each browser afterward or do you test iteratively like after, okay, part 1 is done.

Aaron Gustafson: After a certain stage is done, right?

Lea Alcantara: Yeah, exactly.

Aaron Gustafson: So most of what we do is it depends on the project, really because there are some projects where people come to us and they only want us to build out a prototype or they want us to only build for one particular thing, and

in the case of this Chrome, an app we were building and mainly looking at Chrome and Safari because those were the browsers that the client was specifically targeting because they were building a Chrome app. But in most cases when we are kind of doing our full process, we build our templates as static HTML files and CSS and such. After we build the templates initially, then we do our browser testing and while that browser testing is going on, we usually do our CMS integration or whatever it is that we need to do in terms of the underpinnings of the project from a programmatic standpoint, and then we do kind of a final sweep of testing through all the browsers again once the complete product is done. So the testing usually happens at two different points.

Lea Alcantara: Okay.

Aaron Gustafson: But it's not like we keep all the browsers open while we are building templates. We build to the latest templates, but our process of building and using the faceted design approach to CSS and focusing on good markup and the approach that we've developed over the years that we've been working on the web makes it so that in most cases the progressive enhancement needs are already met as we are building, and we don't end up having that many problems in the older browsers. At this point, I think when we do go back to do one-off fixes for IE7 or 6 or what have you, that maybe we end up with five or ten rule sets mainly to adjust a few minor padding issues or something like that, but not really all that much anymore.

Emily Lewis: Before we close the conversation, I want to take a slight turn from the conversation on progressive enhancement and ask you, you mentioned in your book about this tendency we as developers have to sort of eagerly embrace the shiny new tools that come out and kind of forget what other people might still be using.

Aaron Gustafson: [Laughs]

Emily Lewis: I'm just curious, especially considering the current vendor prefix predicament we find ourselves in, what are your thoughts on what our responsibilities as developers are to ourselves and also to the clients and employers who are pushing for this shiny tools?

Aaron Gustafson: I mean, I'm all for shiny things, but I thing that really we need to think about what the implications of each decision that we make have on the user experience, and are we doing this just so that we can show off to our peers and say, "Hey, check out this awesome thing that I made? Don't you wish you made it?" Or are we doing something that's kind of a nice enhancement?

We just launched a poster contest called *I Love Adaptive Web Design Poster Contest* that Veerle Pieters and Easy Readers, my publishing company are doing, and in the header banner section, there is the heart symbol for the *I Love Adaptive Web Design* and as an enhancement, I decided, "Hey, why not make the heart beat, right, and do some CSS animation stuff?" And so I did that, but I didn't do it just as *-webkit*, kind of with key frames and do the animation that way, but I actually figured it out for each of the browsers and applied the styles in such a way that they would actually work for the different browsers that's support the animations.

In some cases, thinking about the future versions of browsers, so I don't remember if IE10 actually has the animation stuff working yet or not in the versions they've got out, but anticipating that they will probably have a prefixed version in IE at some point before the final spec for animation is released, so putting that in there in addition to the *-moz* and the Opera prefixes and obviously the Webkit prefixes and then the standard version of what it's going to be in the future. I think that's just a smart way to go because even though this page may not necessarily be a high traffic page when IE 10 hits the

mainstream market, if somebody does happen to go to it in that browser, I can give them that nice little enhancement, that nice little visual flourish.

But I think as a web professional community, we need to think about what are the reasons behind doing it? What are the necessities of having that particular feature for all browsers? Is it okay if a browser doesn't get that animation? Yes, that's just a visual flourish on the page. The same thing goes for things like drop shadows or inset shadow or rotation and stuff like that. We shouldn't require that our page rely on those techniques, which are not always guaranteed, or in some cases, do not perform well in terms of animating lots of items on a page that are rotating and so on and so forth via CSS. Some browsers just slow to a crawl with that, which is something that we encountered with our work on the Chrome app that we were doing when we started porting that to other browsers. But to actually brush up on what browsers support which technologies and which techniques, and be aware of maybe the peculiarities of those.

To give an example of that, I mean, for a long time, I don't even know if they've actually made it work yet or not, but you are allowed to put a space in an nth-child selector inside the parentheses on either side of the 2n + 1 or whatever. You can have spaces in there, but if you had spaces in there, at least back at Safari 4, Safari wouldn't understand that selector and so it will ignore the entire rule set. So there are things like that and things like Mozilla's weird interpretation of individual border-radii back when they used to have border-radius topleft with no hyphen between it and things like that. So you have to kind of know what the peculiarities are of the implementations. We can't be lazy in our approach doing this stuff.

Lea Alcantara: [Agrees]

Aaron Gustafson: So if you want to work with the latest fads in CSS and in web design, you need to make sure that you are professional about how you are doing it, that you are not just taking a non-focused approach to doing that sort of stuff.

Emily Lewis: [Agrees]

Lea Alcantara: So I think that's all the time we have for today. Thank you Aaron for joining us.

Aaron Gustafson: Thank you very much for having me.

Emily Lewis: Yeah, it was great.

Aaron Gustafson: It's been great talking to you.

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

Aaron Gustafson: So my company is Easy! Designs at Easy-Design.net. They can read more about my book at http://adaptivewebdesign.info/, or Easy-Readers.net is the publishing company and they can follow me on Twitter. I'm @aarongustafson, and I'm happy to answer people's questions via Twitter. I'm on there pretty frequently.

Emily Lewis: Cool.

[Music]

Lea Alcantara: So now, we would like to thank our sponsors for this podcast, mithra62 and Pixel & Tonic.

Emily Lewis: We would also like to thank our partners, EllisLab, EngineHosting and Devot:ee.

Lea Alcantara: Also, thanks to our listeners for tuning in. If you want to know more about the podcast, make sure you follow us on Twitter @eepodcast or visit our website, ee-podcast.com.

Emily Lewis: And don't forget to tune in to our next episode when we will be discussing selling ExpressionEngine. Be sure to check out our schedule on our site, ee-podcast.com/schedule for more upcoming topics.

Lea Alcantara: This is Lea Alcantara.

Emily Lewis: And Emily Lewis.

Lea Alcantara: Signing off for the ExpressionEngine Podcast. See you next time.

Emily Lewis: Cheers.

[Music stops]