LILY BOND: Welcome, everyone, and thank you for joining this webinar entitled, "Implementing and Evaluating Web Application Accessibility." I'm Lily Bond from 3Play Media, and I'll be moderating today. I'm thrilled to be joined by Jared Smith, who is the Associate Director of WebAIM, a leading provider of web accessibility expertise internationally. Jared is a highly demanded presenter and trainer and has provided web accessibility training to thousands of developers all over the world. He has over 16 years of experience working in the web design, development, and accessibility field and brings a wealth of knowledge and experience to help others create and maintain highly accessible web content.

We have about 45 minutes for this presentation, followed by 15 minutes of Q&A. And this presentation is being presented in conjunction with the Online Learning Consortium. They wanted to let you know about a workshop that they have coming up in September, which you can register for on their website. And with that, I'm going to hand it off to Jared, who has a wonderful presentation prepared for you.

JARED SMITH: Thank you, Lily. I appreciate the invitation and am happy to be here today and talking about web application accessibility. Hopefully you can now see the slides. Just a little bit about WebAIM-- we're a nonprofit web accessibility consultancy based at the Center for Persons with Disabilities at Utah State University. We've been doing web accessibility work since 1999, and our goal is to educate and empower people to be able to build and maintain their own accessibility. I'd invite you to check out our website at webaim.org.

Just setting a little bit of a foundation as we talk about accessibility, I want to maybe define a little bit what accessibility is and maybe how it works a little bit. I think it's important to view accessibility as a continuum. It's a process. It's something that can always be improved. And while that may seem limiting or maybe potentially frustrating-- especially if you're very engineering-minded-- I think it's very empowering to think that you can always be improving your accessibility.

It helps it become a human factor as opposed to simply an engineering or technical fact or implementation. So that means that you can always be more accessible, which, by definition, means it may be less than perfectly accessible to someone. And I think it's important that we accept that so that we set reasonable goals and reasonable definitions of what accessibility is and how we can continually be improving it.

Accessibility encourages good design and development practices. In our work with clients, they continually indicate that by implementing accessibility best practices that it just makes their lives easier. They focus more on good design and user interaction, and even their own coding and development practices are optimized because of that.

It supports search engine optimization. Much of accessibility is machine readability. It's making things compatible to the technologies that people with disabilities might be using-- those assistive technologies. And by making things more machine readable and compatible with those technologies, it's naturally supporting better search engine-- maybe search engine friendliness as opposed to optimization. But we do see a good search engine results and benefits by implementing good accessibility.

Accessibility supports internationalization. Again, we're talking about compatibility with technologies, with a variety of users, and it supports mobile-friendly content. Again, by creating content that's adaptable and friendly to various technologies, that naturally makes it more mobile friendly.

Let me back up. Sorry, here, there we go, if I can get the right slide. About 8 and 1/2% of the population has a disability that affects computer use. That's a very minimal number taken from the US Census data. About 20% of the population has a disability. Of those, a lot of them have disabilities that can affect computer use. This does not include those with color blindness or color deficiency. It does not include those with most cognitive and learning disabilities and so forth.

So you can view that 8 and 1/2% a few different ways. You can view this as, well, by implementing accessibility, we can expand our reach to this 8 and 1/2% of the population. I think there's more to it than that. The web, really, is not all that friendly. It is not all that accessible to people with disabilities. And people with disabilities are going to go to the places that are most accessible to them. And so I really see this 8 and 1/2% or more of the population as really being an opportunity, maybe a market opportunity, to expand your reach and really have this population come to you.

If you don't have a disability, you can easily-- you probably go to the bank. You can go to a classroom to get your education. You can go to the mall and do your shopping. But very often, for those with disabilities, the web really is a primary mechanism for them to engage in commerce and education. And so they're naturally going to go to the places that are most

accessible to them. So that's one way to view this is really as an opportunity for you to reach this market.

But maybe more than that, it's an opportunity to have a significant impact on the lives of people. We can view this as numbers and percentages. But ultimately, this is having an impact on actual people.

Now, if we view accessibility as a continuum, as something we can continually improve, there are measures along this continuum, and these measures are defined by accessibility guidelines or standards. Really, what we recommend, when it comes to accessibility guidelines, are the Web Content Accessibility Guidelines, or WCAG. Or I will say "wickagg." That's how I shorten that is to "wickagg." The Web Content Accessibility Guidelines are a W3C specification.

I suspect that many or most of you listening are probably at least somewhat familiar with them. They are divided into three levels-- A, AA, and AAA. So all of the requirements within these guidelines are assigned a particular level. Level A requirements, if you don't meet those, you almost certainly are going to be excluding users from accessing your web content.

Level AA requirements, if you don't implement those, users will have more difficulty or frustration or take more time accessing your content. And AAA requirements, those are more enhancements to things that maybe are already quite accessibility. It may provide some benefit for more distinct users or groups of individuals with certain types of disabilities, but very often are more difficult or burdensome to implement, but always a good idea. Really, what we recommend and what we have seen as the foundation for legal requirements, both from a legislative standpoint and also from lawsuits and complaints and Department of Justice requirements, have been level A and AA of the Web Content Accessibility Guidelines. And we certainly would recommend that as a good measure of accessibility.

Now, version 2.0 of these guidelines was finalized in 2008. It's starting to show its age a little bit. They're currently, at the W3C, conducting some efforts to update these guidelines. The Web Content Accessibility Guidelines are designed to be fairly principles-based and more technology agnostic. The idea being that by not focusing on technology and being more high level than principles-based, they would have a longer shelf life. They'll still apply as technology changes, which it does a lot.

The side effect of that approach is that the guidelines are somewhat ethereal and confusing. There's some kind of nebulous terms. It sometimes can be difficult to know what they actually mean and whether you are meeting those requirements or those guidelines or not.

Now, there is with the guidelines, the W3C has provided voluminous resources and documentation, many hundreds, maybe over 1,000 printed pages of documentation supporting these guidelines. So that's wonderful. There's great resources out there. But it is quite voluminous.

We do at WebAIM have a WCAG 2.0 checklist on our website that takes the guidelines and those hundreds of pages of supporting materials, and we boil it down to six printed pages, into a checklist that hopefully we have designed to be a little more consumable by folks that maybe aren't real knowledgeable with web accessibility, and they may want to reference that in their development and in their evaluation of web content.

The four core principles of the Web Content Accessibility Guidelines are perceivable, operable, understandable, and robust. These are really good principles to think about as we develop things to be accessible. If it helps us remember these principles, the first letters spell P-O-U-R. We're going to build POUR websites, so that might help to remember these four key principles of accessibility.

So we talked about the four principles of WCAG. But I would propose that before we really think about accessibility, we first need to build things that are useful and that are usable. So we want to build things that people are actually going to want to engage with, that are going to be enjoyable to them, that are going to be informative. And so they need to be useful. It doesn't matter if its accessible if nobody visits it, right? Accessible to whom?

We also want to build things that are usable. It's important that we think about usability early and often as an aspect of accessibility. Usability impacts everyone. Typically with accessibility, we're a little more focused on people with disabilities. And once we build things that are useful and usable, we can then think about building things that are accessible, perceivable, operable understandable, and robust.

So one phrase that I use is web accessibility lipstick on a usability pig. If you have something that is not usable, that's not friendly, no amount of accessibility lipstick slathered on that usability pig is going to make it really usable or accessible. So we need to think about usability first, and then, as we do that, very often, accessibility becomes much easier, becomes much

more natural because it's already usable to begin with. Very often we see things that are not very usable, and the approach or thought is that we'll just add ARIA, or adjustability markup, to it to somehow make it better. And that very rarely works very well.

So your site can be compliant, yet inaccessible. And this is sometimes frustrating. But it's important to realize that compliance simply provides one measure of accessibility on that continuum, but it can still be inaccessible to some users.

So we're always going to favor accessibility or compliance. We're going to favor the user experience over simply meeting a particular standard or technical requirement of accessibility. And to maybe make things even more potentially confusing or frustrating potentially, is that your site can be technically accessible, yet functionally inaccessible. So you can implement technical accessibility, yet still have something that is not very friendly to your end users. So you need to, again, really think about that end-user experience and just how friendly that really is.

So as we dive into these core principles of WCAG and of accessibility, the first is perceivable, the foundational principle of accessibility. We need to get content to the users' senses in a way that it will be useful and meaningful to them. For those with auditory disabilities, this means providing captions for video on live audio and text transcripts for all audio content. So it's pretty straightforward and less easy to actually implement. There are a lot of complexities and nuances to this, but just keep these core principles in mind that captions for video and live audio and then text transcripts for all audio content is what will result in the best user experience.

For those with visual disabilities, which would include blindness, low vision, and color deficiency or color blindness, there are a lot of different types of technologies that can be used-- assistive technologies that can be used by the end user. There are screen magnifiers. There are screen enlargers. The user can increase the contrast or override page colors within their web page. And, of course, for blind users there are screen readers.

When it comes to screen reader accessibility, where the text content of the web page is presented audibly to a user that cannot see that page, there are two primary things to consider and think about. Those are structure and semantics. And some questions that you might ask-- is the application and content structured to facilitate understanding and navigation? So is there really structure there that supports accessibility? And are the

semantics of components meaningful and accurate? So when we say semantics, the word "semantic" means meaning. So is there is a meaning behind the things that we're adding to our web page that can be presented to a screen-reader user?

One of the most important things you can do is provide a good, logical heading structure. Headings are very important to accessibility. At WebAIM we've conducted a series of surveys of screen-reader users, and we found from that research that headings are the predominant way in which screen-reader users tend to navigate and peruse web content, especially content or applications that they may not be as familiar with.

So very important that we have a good heading structure. We recommend typically one first-level heading or h1 per page. That should describe what that page or content or application is or is about.

As you think of defining headings within a page, those headings should define an outline or essentially a table of contents if you were to view only the headings in isolation. You should be able to tell what the content and structure of that document or application is simply by reading or viewing those headings, or for a keyboard or screen-reader users, they can navigate those headings. So headings facilitate not only understanding of the content but also facilitate navigation through that content or application.

So they should be descriptive of page sections for content. That may be a little more straightforward where you would have headings, bigger, bold text that describes sections of content. For a web application, very often there may be components or areas within that application that can be described by headings.

You can also define regions or landmarks within a web application or web content. We can do this in a few different ways. We can use the HTML5 structural elements-- header, nav, main, footer, and aside. Those can all be used to define significant page areas within our content or application-- so the header for the top stuff, nav for navigation. Main is really important. That's for our main content area. Footer, aside is for sidebar information.

ARIA-- and I'll talk a little bit more about ARIA a little bit later-- but ARIA also allows us to define these regions or landmarks by adding the role attribute. So the possible options there are banner for the header-type information, complementary for sidebar information, content info for footer-type information, main for our main content, navigation, and search. Of note, sometimes we get the question of, should I use the HTML5 element? Should I use the ARIA

landmark or both? You could do something like nav role equals navigation.

Anymore, with modern support for these, we generally recommend the HTML5 elements. The accessibility support for those is quite good today. If you're looking at backwards compatibility, maybe back to like IE8, things like that, you may want to use the ARIA landmarks or combine the ARIA landmarks with HTML5.

Of note, however, is that there is no equivalent for the search landmark role. So every page that has a search component probably should have role equals search that defines that search area. Again, these facilitate not only understanding of what are these components on the page, but keyboard and screen-reader users can navigate by these elements within the page.

You can also define your own regions within a page. Many web applications may have panels or components within them that provide certain functionality. But they don't really map well to the available HTML5 or ARIA landmarks. So you can add a role equals region with ARIA the will identify this as a distinct component or region within the page. Every region should be labeled or described with a label, and we can do that with aria-labelledby.

If you're not really understanding the code here, don't worry about it. Just know that this is possible, that you can define a region, and you can give it a distinct label that tells the user what it is. So in this case, the markup describes maybe a filter component within a web application. Filtering doesn't really map to nav or navigation or sidebar type stuff. But it would be important maybe to allow the user to navigate to that component within the application.

So that's a lot with our structure within a page. There's a lot that comes with semantics within a page. We need meaningful links so as the user navigates by linked Items within a page, the links make sense. They're intuitive to the user. They describe the function of that link. So things like "click here" for a linked text are very ambiguous and can be difficult really for all users because of the overhead necessary to determine what that ambiguous link does or what it means.

Alternative text for non-text elements-- very important for accessibility. I could spend the full hour or more, probably much more, just talking about alternative text and the nuance and complexities and rules for alternative text. Just remember two keyword-- content and function. What is the content of an image, and if the image has a function, what is that function? Those are the things we want to convey in our alternative text. And we have articles on the WebAIM

site for all of these things that I'm going to be talking about and can help you get a little more detail and insight into how to implement them.

Labeling form controls-- visually we can look at a piece of text to say the word "name," and next to it is a text box. And we can visually create an association between those. We know we type our name into that text box. Because of that association and proximity of the word "name," someone that's blind cannot make that visual association, and so we need to provide an explicit association that says, this field here is for the user's name. And the screen reader can identify that label if its a properly associated when that user navigates to that form control.

Button values are also important. Buttons are very critical to functionality on the web. So they must have something that describes what that button does. Very often we'll see things like CSS background images defined for buttons, but the button does not have a value or text that is in the content, in the markup, that defines what that button does. Those become very difficult for a screen-reader user, if it simply reads "button," if there's no text to describe what that particular button does.

Also need to associate data cells to row or column headers. So if you have a data table, you can visually scan up or down, left or right, to determine what a particular piece of data is or what it is, based on those headers. Someone that's blind can't visually scan, but we can provide a programmatic or explicit association in our markup by defining our table headers and defining whether those headers are column headers or row headers. That then creates an association between those headers and the data within those columns or headers, and those relationships can be identified to a screen-reader user as they navigate that data table.

Page titles are also important for semantics. That helps define what the page is about and is one of the first things read by a screen reader. Defining the document language is also very important. Especially for multilingual users, it helps ensure that that page is read with the proper language settings and intonation and so forth by the screen reader-- very easy to do.

Most of this is quite basic markup that we can add into our web page. It's under the hood of our web page, generally doesn't impact the functionality or the visual presentation. It simply is making that content more accessible, primarily to screen-reader users.

Now, as we talk about semantics or meaning of elements, it's important to remember that HTML elements have default semantics. Buttons have a default semantic or meaning of

button. Links have semantics of link. Tables have semantic information that says, hey, I'm a table. So keep that in mind. We want to use HTML elements appropriately so that those semantics are conveyed to users appropriately, so they can then know what a particular thing is that they're reading or interacting with.

Occasionally, however, especially with more complex things we may see in web applications, sometimes the semantics of HTML are not sufficient. Maybe we're creating something for which there is no HTML element that helps define what that thing is. In those cases and in only those cases, then you can use ARIA. ARIA is Accessible Rich Internet Applications. It's a specification by the W3C that helps us extend the accessibility of web content, particularly for web applications, to provide better accessibility.

However, there is an important rule. The first rule of ARIA use is, essentially, don't use ARIA. What it actually says in the spec is if you can use a native HTML element or attribute with the semantics and behavior you require already built in, instead of repurposing an element, such as a div or a span, or maybe some other element, and adding an ARIA role, state, or property to make it accessible, then do so.

So what this means is if you can make it accessible using HTML, then you must. That's really what you should do is start with the semantics of HTML. When that is not sufficient, then you can use ARIA. Unfortunately, while ARIA provides great potential and benefits for accessibility, because of abuse and misuse and overuse, unfortunately, it seems to be resulting in web content that is less accessible.

In our work with most of our clients, where we're helping them with accessibility of web applications, we tend to spend more time telling them how to stop using ARIA than how to start using ARIA to enhance accessibility. Very often ARIA is added when it's not necessary, or it's added incorrectly, and that can very, very easily destroy the accessibility of a web application. Even though it's [AUDIO OUT] to enhance, it very easily can make things totally inaccessible to screen-reader users simply by adding one attribute to a particular element.

So really what I'd say regarding ARIA is if you're going to implement ARIA, any time you type ARIA dash into your markup or any time you add the role attribute, please, please make sure you're doing it correctly. While your intentions may be good, if you do not do it correctly, it generally will make things less accessible or potentially totally inaccessible to your users. So with great power comes great responsibility. ARIA definitely fits into that category.

So really what ARIA does is it changes and enhances the default semantics of HTML elements to API values, which screen readers already understand. So when you have markup within a web page, the web browser interprets that markup and passes through accessibility APIs information regarding those elements in a way that the screen reader can understand that in a standardized way. So when you encounter a link, for instance, the browser interprets that and passes on an API value or flag that tells the screen reader, hey, this thing is a link. This is the link text, has it been visited or not, those types of things. And the screen reader can then read that appropriate information.

ARIA allows us to add markup to our page that adds to those semantics that are defined in HTML so we can do new and cool things. A good example is maybe a slider within an application. We don't have a slider element in HTML. There really are no available semantics for slider. But we build them quite often in applications. ARIA allows us to define the semantics and values for sliders-- the current value, the minimum value, the maximum value, things like that-- in a way that can be defined via these APIs so that screen readers can identify what those are [? to ?] be accessible.

Very often our recommendation when it comes to accessibility can be summed up in this one simple statement-- just use a button. Use the standard HTML element. Very often we see maybe what we might call pseudo buttons in web applications.

They're a div element or a span element that has the onclick event handler and maybe additional markup to make it keyboard accessible. And if you do that, then you must also make sure that it responds to keyboard interaction as well as the mouse. And that turns out to be really quite tricky, and it works differently on different browsers. So you're doing like key event detection.

And then the semantics aren't really there for a div or a span that [? annotates, ?] hey, this thing is a button. So you add ARIA to it to add those semantics, which kind of is a violation of that first rule of ARIA because there is a button element in HTML that provides the necessary accessibility.

So, again, this goes back to that fundamental rule. If you can do it with HTML, you must do so. And very often what we'll see in one of our, say, inaccessibility report that we do, we'll see one of these fake, pseudo buttons. And the recommendation will be use a button. If you don't do that, here's a page and a half of documentation on what you have to do in order to ensure that

this pseudo or fake button actually is accessibility. Certainly, creating and just using a button is much, much easier and much more accessible.

A similar thing would apply to other elements. A very common case that we're seeing are select menus, where instead of simply using a select element in HTML, which provides the keyboard accessibility and the proper semantics, many developers are creating these pseudo select menus with a bunch of other things built into them. So they're much more difficult to make accessible when HTML will do the job.

A few other things to consider for users with visual disabilities-- you want to provide sufficient contrast to the user. There's guidelines or thresholds in the Web Content accessibility Guidelines that can be used to help define what is sufficient contrast. You want to ensure that meaning is not conveyed with color alone. Users may override color values within a page. They may turn on high-contrast mode.

They may have color deficiency, where they can't see or perceive certain colors. And screen-reader users are essentially color blind. The screen reader would not generally identify the color of elements within a page. So there are a lot of users that can be impacted by relying on color alone to convey information or meaning.

You also will want to ensure that content is operable, our next principle of accessibility. Generally this will affect users with motor disabilities that may use a variety of assistive technologies to interact with and navigate within web content. So we want to make sure that things are keyboard accessible. A user may not be able to use a mouse and may be interacting with the keyboard or other type of technology.

We want to make sure as a sighted keyboard user is navigating through a page that they can tell which thing currently has keyboard focus. They should be able to see as they hit the Tab key and navigate through interactive elements, they should be able to know what thing they're currently on and that they can interact with or activate. These keyboard focus indicators can be inhibited or turned off with basic CSS of outline 0 or outline none. So you want to avoid that to ensure that it's accessible to sighted keyboard users.

You want to make sure that things that are interactive are clearly distinguishable. And this is helpful for everyone. You want to know with what can you interact with on the page. If a user reads or navigates through a page, the order of the content that's read or navigated should be logical. Generally that means it will follow the visual order through the page. This becomes a

little more difficult with complex web applications based on layout. But we need to ensure that order is logical to the user.

With more complex interactions within a page, very often we have interactions that can draw attention to certain things. Say the user clicks the button and it triggers a dialog window. We need to set programmatic focus to follow that visual focus. So when that dialog opens, we would need to use scripting to set focus to that dialog so that the screen-reader user or a keyboard user would immediately begin interacting with that dialog.

When a user closes or dismisses that dialog, we would then need to set focus back to a logical place within the page. So much of web application interaction is going to be setting focus to make sure that programmatic focus, or where the focus is set for a screen reader or for a keyboard user, follows that visual focus. You can also for complex navigation or repetitive elements within a page, you can provide links that allow the users to skip over those repetitive or lengthy lists of links.

This can be helpful for web applications, where you can provide links maybe at the beginning of the application so the user can easily navigate to various panels or components within that application. And you also want to give users control over time-sensitive changes. When things are changing without user interaction, maybe automatically, there's the potential for confusion, especially for someone that can't see the visual presentation-- maybe somebody that reads very slowly. As much as you can, you want to give the user control over those content changes within a page.

There's a lot that you can do with interoperability with ARIA to help enhance accessibility. You can use aria-expanded to indicate the state of elements that it spanned or collapsed. And very often we see like these accordions or a plus or a minus or a caret symbol that indicates whether something is expanded or collapsed. We can indicate that to users that can't see that visual presentation.

aria-haspopup equals true would indicate to a screen-reader user that activating this element will trigger a dialog or a menu or a pop up, as opposed to a link that would generally take you to another page or a button that would usually perform a function or submit form information. Very often we have these things that are kind of in between links or buttons. They don't function really as a link but don't really function as a button. They perform an in-page function, such as expanding or collapsing or triggering a pop up.

So ARIA allows us to add those semantics to enhance accessibility. aria-live allows us to define what are called live regions. Live regions are areas within a page that may update dynamically, and this allows us to make those accessible and gives us control over how the content within that dynamic element within a page would be read by a screen reader. There's aria-required for defining required form controls, aria-invalid for defining invalid form controls-- form controls for which an error has occurred, and the user must address or fix that error.

There's role of alerts to trigger important, very vital information that you would want to be read by a screen reader without necessarily setting focus to that content and so on and so forth. This is just the tip of the iceberg when it comes to ARIA. There's so much out there that can enhance accessibility, beyond what we can do with standard HTML.

As noted before, be careful. It's very easy to make things less accessible if you don't implement these correctly. Fortunately, the ARIA specification is really quite well-written. It's very descriptive. There are many examples out there that you can follow to ensure that you're implementing ARIA correctly. And, of course, you probably want to test it, and we'll talk about testing in just a moment.

You want to ensure that content is understandable, our third principle. The things that we're conveying are accurate, that they're understood correctly by the user. This will affect users with cognitive or learning disabilities. This is by far the largest disability group. And really it affects everyone. We all benefit from having content that's easily understood. So we need to be careful with movement or other distractors. A very basic animation or video within a page might be mildly annoying to you, but it may render the entire page content totally inaccessible to some users that can't ignore that animation or movement.

We talked about good organization. It supports good content understanding, using HTML and semantics of headings and lists and tables appropriately. Simplifying and being consistent in our presentation. Small text-- we conducted research at WebAIM, and we found that small text-- not super tiny, but smaller text-- really impacts users with cognitive and learning disabilities, and I think it impacts everyone. Long line lengths, where a lot of characters per line can impact readability.

We want to focus the user's attention on the things that are most important, and that's going to be the content within our page. We can chunk or break up complex content. We can simplify it. A lot of what we're talking about is balancing cognitive load and functionality.

If you think of your home page, for instance, home pages tend to have a lot of stuff. There's a lot of functionality. There's a lot of content all packed into a home page. So there's a very high cognitive load, but there's also a lot of functionality. How much can we decrease that cognitive load while maintaining the critical or important functionality? This requires difficult decisions. But it really impacts accessibility for everyone.

Any time that you have to think, any time that you're confused or experience frustration or you have to remember something to interact with, say, a form or an application or content, magnify that frustration or effort or that cognitive load by 1,000. That may be the type of interaction or experience that a user with a certain type of cognitive or learning disability might have on your web content.

Then our final principle is robust. Robust deals with technology strength, making sure that things actually work in the technologies that your users have. And there's not a lot in the Web Content. Accessibility Guidelines. Well, there's two success criteria or requirements there. But there's actually quite a bit that's built into them and that general principle of just making sure that you're following the rules and that things are actually working for your users with what they bring to their table.

So those are our four principles that provide good guidance for accessibility, generally. If we think about these from a human perspective, thinking about the different types of users with disabilities, it helps support good accessibility. We also want to consider age-related processes. As we age, we tend to lose visual function. We tend to lose hearing. We tend to lose motor function. And we lose our minds. We tend to lose some cognitive function as well.

This is one thing that really motivates me and keeps me working in this field is that I know that by making the web more accessible symbol today, I'm making it more accessible to my future self. Statistically, most of us will experience some form of disability within our lifetimes. And so we're all making the web better for our future selves, and that helps motivate me.

Finally, a few words about evaluating. Really, we've talked about principles of accessibility, and implementing those principles in development is important. But those same principles would be implemented in evaluating. The Web Content Accessibility Guidelines are designed to be evaluation standards. And so those four principles and the techniques we've talked about, we would also want to check those in our evaluation.

It's important to note that only people can evaluate true accessibility. Accessibility is about the human experience. You have to be a human. Tools can be useful. They can identify some components of accessibility. But human evaluation will always be necessary.

So some things you can do. You can use a checklist-- I talked about the WCAG 2.0 Checklist that WebAIM has or WCAG itself-- just to ensure that you're addressing these critical aspects of accessibility. Keyboard testing is very important. The prevalence and significance of keyboard accessibility issues has significantly increased in recent years. If you would have asked me a decade ago something that I thought we would have had figured out by 2016, I would have probably said keyboard accessibility. Unfortunately, that's not the case, and keyboard accessibility is getting worse. And much of this is because of complex web applications.

So there's much that we can do there. Fortunately, keyboard accessibility is easy to test. Put your mouse away. Using only the keyboard, can you interact with all elements within the page, and is it functionally accessible? Is it efficient for you using the keyboard to interact with that content? And it's not all that difficult to implement. It just tends to be very often overlooked.

Test in a screen reader. Screen-reader testing is more important as you get more advanced or complex web content. As you begin to implement, say, HTML5 or ARIA, then screen-reader testing becomes more necessary. While screen readers are fairly complex pieces of software, they're not all that difficult to use for basic testing. We have articles on the WebAIM site to help you get started with three of the more common screen readers-- JAWS, which is the most prominent screen reader; NVDA, which is a free open source screen reader for Windows; and VoiceOver, which is the screen reader that comes with Mac and iOS devices.

User testing, very important, does not have to be very complex. But if you want to ensure the content is accessible to users, getting users to help in that testing is very important. You can also use automated tools. With the understanding of their limitation in helping you understand whether something is actually accessible, it can be helpful in monitoring and helping to identify accessibility issues.

We at WebAIM developed the WAVE tool. It's freely available for you to use. It's at wave.webaim.org. So I'd invite you to use that. You can go to the site, type in your web page address, hit a button, and it will visually present information regarding the accessibility of that page.

WAVE has been structured and designed to help facilitate human evaluation. We reveal underlying accessibility information to make it easier for you, as a human, to determine how accessible something is. We also have a Chrome extension that is available, so you can perform the evaluation within the Chrome web browser.

I just want to conclude with this quote. "For people without disabilities, technology makes things convenient, whereas for people with disabilities, it makes things possible." It's easy to think of accessibility as these techniques and markup and code and using tools and checking checkboxes and checklists and meeting accessibility requirements.

But I hope you understand the real potential and possibility that the web has for people with disabilities. It's so enabling. It's so empowering. And you have the potential and the opportunity, by implementing accessibility, to very positively impact the lives of many people with disabilities. So that's my charge to you, is to continue with accessibility. And with that, I will say thank you, and we'll go back and see if there are any questions.

LILY BOND: Thank you so much, Jared. And that was a really nice note to end on. There are several questions coming in, and I want to encourage people to continue to ask them. As I'm compiling the first few, I wanted to let everyone know that we have a few more upcoming webinars-- a webinar on July 29 about Yahoo's Corporate Accessibility Strategy. And then in August we have the impact of recent lawsuits on video accessibility. In September, we have Quick Start to Captioning. And at the end of September, Lainey Feingold, who is an internationally recognized disability rights lawyer, will be presenting on legal updates for digital access cases.

So, Jared, to start off, someone is asking if there's a particular font and font size that you recommend for websites?

JARED SMITH: Well, that's a good question. When it comes to font faces, one thing we've found is that most good research out there, modern research, tends to show that it doesn't matter a whole lot when it comes to most accessibility, readability, retention, things like that. So I don't think there's a particular specific recommendation-- use this font face. Just use good font faces that are generally readable to the user.

We see old recommendations of only use sans serif fonts for print and serif with the hooks and flags, like Times New Roman for print. Those are really quite dated. With modern displays and resolutions, that doesn't seem to have much of an impact anymore. So if you build your corporate website with Comic Sans Serif, that may not really impact readability, but it is going

to impact perception of the site, which I guess could impact accessibility and usability.

As far font sizes, I haven't seen anything specific. Too small-- you know it when you see it. But if you want a number, anything less than about 10 pixels tends to become more difficult. That's what we found in the research we conducted with users with cognitive disabilities.

LILY BOND: Thanks, Jared. Someone else is asking, what are practical ways that someone, like a developer, can assess the end-user experience and favor the experience over simply meeting compliance, and how do you evaluate that?

JARED SMITH: Yeah, you know, that's difficult. I think that's kind of what accessibility is about. I guess I would say here that much of the difficulty with that question is that the people that are designing and building and maintaining web content for a particular site are so intimately familiar with that because they built it, that very often it's difficult to see those usability types of issues on their site-- so I think getting other people to take a look at, that user testing.

I wrote a blog post on our website. It's been a few years ago, but it was about web accessibility testing, user testing with people with disabilities, focusing on that broader user experience, as opposed to simply identifying accessibility issues that may provide some insight there. So stepping outside of your own experience with a website and doing that type of user testing is probably the most important thing that you can do there. These are the types of things that can't be evaluated by a tool or with a checklist.

LILY BOND: Thanks, Jared. Someone else is asking, where is WebAIM located, and do you provide face-to-face training?

JARED SMITH: We're located at Utah State University in Northern Utah. And, yes, we do. We can come on site to provide training. We also host trainings here in Utah four or five times a year. There's information on the webaim.org site about our training offerings as well as other services that we can provide.

LILY BOND: Thanks, Jared. Someone else is asking, did you say that people with cognitive disabilities are the largest group or blindness and low vision the most common?

JARED SMITH: Yeah, those with cognitive and learning disabilities, as near as we can tell, it tends to be a little difficult to get really good numbers and statistics about disability, especially with web content and web interaction because you can't really detect whether a user has a disability on your

website. You can detect whether they're using Internet Explorer or Chrome. But you can't tell whether they're using a screen reader or a screen magnifier or if they're navigating with the keyboard, those types of things, at least it's difficult. And then I'd ask, even if you could detect that, what would you do? Hopefully, you would still focus on universal accessibility.

Yeah, while it's difficult to get good numbers, as near as we can tell, those with cognitive and learning disabilities likely is larger than all of the other disability groups put together-- larger than those with visual disabilities, hearing disabilities, and motor disabilities put together. Upwards to 20% of the population, I've heard numbers that suggest that. So it's quite significant.

LILY BOND: Thanks, Jared. Someone else is asking, do you have any good resources for ARIA usage?

JARED SMITH: Good resources for ARIA usage-- probably the best resource I would recommend is the ARIA Authoring Practices. I'm pretty sure that's what it's called is the ARIA Authoring Practices document. Within it, there's something called design patterns, and the ARIA design patterns are awesome.

Basically what they do is they say, if you're building this type of component, say, a dialog window or a slider or an expanding and collapsing component, you can pick the type of the component or element that you're building, and it will document. It was tell you, this is what this thing is. This is what the keyboard interaction needs to be that you need to define so that user is interacting with it in the standardized way.

This is the ARIA markup that's necessary for it to be assessable. And then it also references several examples of those widgets or controls that have already been built, so you can go and look and see how they're built. So that's one good resource. We have an overview, a pretty high-level overview, of ARIA on the WebAIM site as well. There are quite a few other examples out there.

LILY BOND: Thank you. There are quite a few questions about title and h1 usage. So I'm just going to group them together. Someone is asking if you could talk about the difference between a title and an h1 and how many headings are too many? Should we only use one h1 per page? And wouldn't one use the title in brackets for the main title and then h1 for top level headings, like Roman numerals in a formal outline?

JARED SMITH: So the title element in HTML, that will define, essentially, a description for the document itself.

That is what shows up in the blue bar at the top of your browser within the tab, within tabbed browsing. That is read generally by screen reader when the user comes to that page or if they switch windows or tabs. So that should be descriptive of the page content.

The h1, or first-level heading, has a very similar function or purpose. Generally, it should define what the page is. They may even be very similar or even identical, the title and the h1. But they're a little different in how they're presented and their functionality. And so even if there is maybe some redundancy there between them, a screen reader may even read them twice-- once when the user comes to the page and then again when they get to the h1. That's OK.

The h1 is used to define, within the document itself, what it is and defines that root level within the outline. We would then define subheadings below those h2s and h3s and so forth to do subsections within that. So hopefully that kind of answers that question. Again, we do have an article on the WebAIM site. And the question of how many is too many? That will vary based on the complexity of the content.

Any heading is better than no heading at all, if it's truly a heading. So make sure you're marking things up as heading. You can have too much of a good thing, but rarely the opposite is the case. We see things that should be headings that are not marked up as headings.

LILY BOND: Thanks, Jared. Someone else is asking, are there any courses and testing that a developer can take for an official certification in web accessibility?

JARED SMITH: There are a few that are out there for accessibility certification. There's a relatively new one by the IAAP, the International Association of Accessibility Professionals. As a new organization, they have recently rolled out a certification for developers. I believe the NFB also has a developer certification. So, yeah, there are some out there. The question is really the current value and applicability of those in the field itself because this is a relatively new field. But more than anything, regardless of the certification or which one you go for, gaining that knowledge is really important.

LILY BOND: Thanks, Jared. We have time for just a couple more questions. Someone is asking, what resources could you provide for low budget companies and organizations to do accessibility compatibility testing?

JARED SMITH: What resources-- accessibility, I think, is not all that difficult if you follow the standards, if you

follow the guidelines. That idea of accessibility compatibility testing is kind of interesting. When we think about compatibility, I think of like does it work for this particular browser or this particular screen reader? And usually, if you follow standards, you don't have to worry too much about that because of the support in the various browsers and assistive technologies is good.

There are a lot of free tools that are out there-- certainly using those. But really, when people ask us how much does web accessibility cost, it's always a factor of knowledge and education. The more you know about accessibility, it just becomes a natural part of what you do. And so things like this webinar, WebAIM, the great work that 3Play Media is doing, and others helps inform accessibility and makes it much easier.

LILY BOND: Thanks, Jared. Final question, and sorry to everyone that we couldn't get to, are there any accessibility guidelines for mobile devices, cell phones, and tablets, or is it an extension of the general accessibility guidelines for web?

JARED SMITH: Great question, for mobile, that is an area that the W3C is working on. There is an article or a document out there that talks about the applicability of WCAG 2.0 to mobile that can be insightful. They are looking at updates to WCAG 2.0. They're starting to define these more concretely. So the W3C has some really good resources. For the particular authoring platforms, for iOS or for Android, there is guidance there. There's details on how to implement accessibility, at least in native mobile apps. So that's also a good resource.

All of that's a little bit new. As far as like technical guidelines and standards, a lot of that is yet to come, as far as like official W3C specifications for mobile accessibility. We have to use a lot of other things and kind of extrapolate from those to get to a guidance for mobile, as of right now.

LILY BOND: Well, thank you so much, Jared. That's, sadly, all we have time for, but thank you so much for a really great presentation and the really valuable Q&A, too. So thank you so much for being on the line today.

JARED SMITH: Thank you for having me. I appreciate it.

LILY BOND: And thank you to everyone who attended. A reminder that you will receive a copy of this presentation tomorrow, along with the slide deck and the full transcript. And thank you again to Jared, and I hope everyone has a great rest of the day.