SOFIA LEIVA: Thanks for joining this webinar entitled Toolkit for Digital Accessibility. I'm Sofia Leiva from 3Play Media, and I'll be moderating today. And today, I'm joined by Jack Nicolai, accessibility product manager at Adobe. And with that, I'll hand it off to Jack, who has a wonderful presentation prepared for you all.

JACK NICOLAI: Greetings. My name is Jack Nicolai, accessibility product manager for the Creative Cloud at Adobe. You can download this presentation deck in both PowerPoint and Word document format from Dropbox using the bit.ly link http://bit.ly/accessibility-toolkit, along with a set of assets that I'll be introducing in the presentation.

So today, I'll be talking about a problem I identified while working with software development teams. I'll be proposing a solution to that problem, and talking about who's involved in the solution and how we go about addressing it.

I interact with a lot of people who know little to nothing about accessibility but are being tasked with making their applications accessible. Until we have a certain level of fluency in accessibility, I tend to document accessibility requirements in detail. This is in part to educate and to anticipate what questions may arise. Ultimately, the point of this presentation today is to get us talking.

So the problem, as I understand it, is that accessibility requirements aren't documented clearly, consistently, or in a way that other professionals can easily understand and act upon them. So the solution, really, is to employ standard documentation methods to express accessibility requirements, which can be written by professionals knowledgeable about accessibility, understood by stakeholders, actionable by engineers, and used as a basis to validate functionality.

And in that process, we want to create artifacts that document accessibility requirements, which then drive conversations about how to make your software accessible. I have to say, it's a lofty goal, which I haven't always fully achieved success. What I share with you today are things that I commonly do in my own work, and I've seen different versions of from other accessibility professionals that I think will be of value to you.

So who are the people that are actually supposed to be authoring accessibility requirements? In my experience, it starts with the product manager, as they're defining the features and

requirements for a particular piece of functionality; experienced designers, as they are thinking about how different users will actually be interacting with these features or software; graphic or product designers, as they're creating a UI visually to represent those features in a way that people can understand and utilize; content strategists that will be writing both things like labels as well as informational instruction, that sort of thing, that users will need to be able to properly use the application; and then other types of accessibility professionals that may be contributing their knowledge to the process.

So this is a representative list, which could include additional roles. I would like to conduct a quick poll to find out who in your organization is documenting accessibility requirements today. And then we'll come back to that in just a few minutes, when we've got some results that have come in.

So who are the consumers of these accessibility requirements within a product development team? Certainly, the experienced designer, as they're getting requirements from the product manager. And then it begins to roll down through the development process, to the graphic and product designers, the same content strategists or writers. Often, they are collaborating with the product manager and other team members to figure out what the user experience is going to be, including those individuals that are utilizing assistive technologies or perhaps the keyboard alone to navigate the software.

Those requirements then filter down to the engineers and the testers, so that the engineer understands how to build what they need correctly to be more inclusive and take into account those that are using assistive technologies, and the testers to be able to actually write tests that actually validate that those things are working, business stakeholders that want to understand how it is that we're meeting the needs of customers with disabilities, and then other accessibility professionals as well.

So communicating accessibility requirements. We'll be looking at a few methods of expressing accessibility requirements that will help to address the Web Content Accessibility Guideline principles of perceivable, operable, understandable, and robust. In particular, we'll look at a set of graphical assets, graphical elements, rather, used that help communicate accessibility requirements in wireframes or design comps, and how those assets can be included in user stories.

So consider, what are the formats that you use to communicate accessibility requirements

today? The ones that I find that are most commonly utilized are user stories, wireframes, design comps, as I've mentioned, design specs or design patterns, within technical specifications, working prototypes, and then finally, in test cases.

So we'll be looking at three different methods today in which to deliver accessibility requirements-- user stories, design specs, and test cases. I feel like these are the core places that accessibility requirements need to be expressed and addressed. Some of the others that I mentioned can be tertiary and also ways of including accessibility requirements. But really, these three methods are where they start within the development process.

So one of the key requirements to communicate is the order in which a user interacts with an interface. This is a common interface used to search and filter a set of information. In this example, I'm searching for restaurants near me. Along the left rail is a set of collapsed accordion-style categories of filters labeled Neighborhood, Cuisine, and Price.

In the main content area to the right of those filters is a set of search results displayed as a grid of cards. Each of the cards summarizes the result and may include related links or other calls to action, like to make a reservation. So there's a toolbar above the results, which allows you to sort the results by either rating or alphabetically. And then below the set of search results is a navigation region in a numbered pagination format, with a Next and Previous link to be able to then view additional search results.

So we're going to be taking this wireframe of an interface and using these three different methods to show how user stories, wireframes, and test cases can be used to describe the accessibility requirements needed for this particular interface.

The first one that we're going to be taking a look at is our user stories. And many of us don't use them actively today, and so I'm going to take just a moment to describe what they are, and then we'll take a look at an example. So a user story is a high-level, user-centered definition of a requirement, which ultimately is to deliver value, containing just enough information so that the developers can produce a reasonable estimate of the effort to implement it, and tests can be written to validate it.

So the format, typically, used to describe a user story is as follows. So as a particular type of role-- so let's say, as a keyboard-only use-- I want to navigate a particular interface to achieve a certain benefit. So as a blank role, I want to complete some sort of goal or desire so that I can achieve a particular benefit. Stories may include additional information and resources,

such as additional context, acceptance criteria, diagrams, technical specifications, and links to other resources.

Ultimately, there are a lot of different ways that you can slice functionality. Typically, user stories encompass a complete slice of functionality. But for the purposes of this presentation, I'm slicing the functionality into a few more discrete pieces, really for the purposes of clarity and a more discrete test definition. So it's not really an all-or-nothing situation. Take from this example the elements that are useful to help you define and test the requirement.

So let's take a look at an example of a user story that addresses a particular set of accessibility requirements. So we have here the title of the user search, Keyboarding for Search Results, and the description. "As a keyboard-only user, I want to keyboard navigate and filter the search results for restaurants near me so that I can find a place to eat. Focusable elements should be in a logical order and display a clear indication of focus."

So already, we're seeing in the description for the user story both the context of who the user is, what is the goal that they have to achieve, which is to find a restaurant near them, and some of the things that are important within the experience for them to be able to do it successfully. I would suggest checking out Kathy Wahlbin's article, "How to Write User Stories for Web Accessibility." If you download a version of the deck, both in PowerPoint or a Word presentation, the link is included in the speaker notes.

So let's take a look at the next portion of the user story, which would be the acceptance criteria. So again, this user story is being written for that search results interface that we were just looking at a moment ago. So just to remind you, on the left of the interface was where these accordion widgets for things like-- to filter which neighborhood that you're looking at. So the first acceptance criteria is, all functionality of the content is operable through a keyboard interface. It's a blanket acceptance criterion. And then we'll begin to break down, then, into some of the more specific requirements here.

So the Tab key. The next acceptance criteria would be that the Tab key moves through the list of search results in the natural keyboard order of the Document Object Model. Now, the context of this interface is that it's within a web page. And so for those folks that are familiar with web development, you should at least be familiar with the Document Object Model, which is essentially a description of the interface, but on the programmatic side.

So the next acceptance criteria would be, with a focus on a filter heading, the Space or Enter

key will expand the filter accordion. The elements inside an expanded filter should then be added to the tab order in a manner indicated in the associated diagram. So this acceptance criteria references additional resources that would be associated with this user story. And we'll actually be looking at one of those design specs or diagrams next.

So when it focuses on a filter heading, then the right arrow would expand the accordion and the left arrow would collapse the accordion. When it focuses on one of the children of that accordion, pressing the up or down arrow key will move focus to the next or previous filter in the list.

So hopefully, you can see that we're getting down, but-- we're both describing a feature at a macro level, and then also breaking down, what are the individual expectations that I should have as a keyboard-only user to be successful to use this interface to complete the task that I have, which is to find a restaurant near me?

So while there are a number of different things which can be communicated regarding accessibility, at minimum, you want to communicate how someone gets to an element in the interface, the element's label, what kind of role it plays, whether it's, let's say, a button or a link, what sort of states and properties it has-- in the case of the accordion interface, it's whether or not it's expanded, so that you know whether or not you can then navigate into the children of that accordion, and then when it has a focus and how to interact with that element.

And finally, I talked about how additional context can be included in the user story. And when I think about handing off this user story to an engineer, I want them to have all of the essential resources that they need to be able to understand how to go complete this particular set of work.

And if I've got an engineer that's not particularly educated about accessibility requirements, especially within web context, I want to point that individual to the Web Content Accessibility Guidelines that are relevant to this particular slice of functionality. And so here, as context, I'm included links directly to reference material by the W3C for these particular guidelines.

In addition, there is a specification called ARIA, or Accessible Rich Internet Applications. Sorry, I'm forgetting that acronym fully right now. But it's basically a set of descriptions and design patterns for all kinds of different UI elements including one that we were looking at within this user story, which is the accordion pattern, or within the search results itself, which follows a

grid pattern, design pattern, as well as the toolbar for filtering either by rating or alphabetically.

And so I'm basically packing into this user story all of the reference material, essentially, that that engineer and eventually the tester are going to need to be able to successfully write both the test cases and the actual code to implement this feature.

OK, so I'll talk next about how we're going to take a look, essentially, at a diagram or a design spec that you would include along with the user story to support it and to help describe the functionality. So we're going to be taking a look at a method that internally here at Adobe we've begun to call bluelining.

And when you're thinking about, from a design perspective, the term "redlining," which gets down to the nitty-gritty details about an interface, things like the pixels between different elements, it gets really quite specific. And what we want to do here is, at the design phase, also be describing, what are the things that need to be expressed in this design so that an engineer and a tester can really understand how to implement it? So let's take a look.

There's two key things that I want to make sure that get expressed, or key concepts to annotate within these design specs. One is a concept called wayfinding. And within that, I consider it to include things like the focus order, whether that be through keyboard, or if I'm using a screen reader, let's say on a mobile device, I might be swiping back and forth to change the focus.

So what is that keyboarding experience? What occurs when someone uses the Tab key? What is the focus order? Are there keyboard shortcuts that need to be expressed and available to users? And what kind of content structure is there?

And then finally, I like to think about and express, essentially, what I call the content behind the content. And often, I may see an interface where there are elements in the UI that don't have a visible label. I might have a row, a toolbar, for example, that is a set of icons.

Now, for someone just visually looking at the interface, they may derive enough information to be able to understand what to do with those elements. But we still need to plan for and programmatically associate an actual textual label to describe that element, so that if someone is utilizing that interface with an assistive technology, there is still a label to express to, let's say, a screen reader. Additionally, elements or pieces of information like that element's role, its state, and other properties also need to be basically decided early on so that an engineer then

knows what to implement.

And then finally, sometimes, there can be additional content that we may find is important to express to someone, especially someone who is not seeing the interface, that may be needed to provide additional context so that they understand where they are and what they need to do. So we're going to take a look at some examples utilizing this set of what I refer to as accessibility annotations, and that it covers these concepts of wayfinding and the content behind the content.

So we have elements that help to define tab order, annotations, or that content behind the content that assistive technologies may benefit from, things like keyboard shortcuts, a general icon for additional annotations, annotating different regions of the content. And within a web context, we might think of ARIA landmarks, for those that are familiar with them.

But even within a desktop application, it can be useful to define different regions. Let's say you have something like a mail application, and having a region that clearly defines where messages should appear versus where a list of the directories might subdivide or organize all of your messages into. And then, really simply, just letter keys to help to define the keyboard shortcuts. And as well, we'll look at, basically, how directional arrows can be used to help quickly and easily describe how a user moves through an interface.

So I created this set of graphical assets in Illustrator. I chose Illustrator because of the various image formats that I can export, including SVG, and the support to copy and paste these assets as vector art into other applications, including Adobe Experience Design and Sketch. I could also include these assets in a Creative Cloud library so that I can easily share them with other Creative Cloud subscribers.

Additions could include notations for swipe gestures for a touch interface, for something like voiceover or talk back on Android, and for Windows 10. So let's actually take a look at an example of some of these annotations put to work.

So we're taking a look at that same search results interface, but in this example, I'm looking to illustrate what is the keyboarding experience-- and, in particular, the focus order. How is the user going to move through the-- how can they expect, essentially, to move through the interface with the Tab key? And so in this diagram, I've used the notation for the Tab stops to basically numerically mark up the interface so that someone can quickly look at this diagram and understand, OK, so where do I start?

My first Tab key should start at the first accordion filter on the left for the neighborhood, and then move down through that list, and then over to the toolbar, and then down into the search results, and then finally, into the pagination controls. So along with that diagram, I'm including some of that same acceptance criteria that showed up in the user story. So for example, a Tab key moves through the list of search results in the natural keyboard order of the Document Object Model.

So very clean, very simple. But when I hand this off to an engineer, they can, within moments, look at it and say, OK, yep, I understand. This is what your intent is, and you've included things like what do different keys do. Great. I can go build that.

So then let's take a look at a more detailed example. So we want to first start by looking at, how does someone move through the interface as a whole? And then we want to actually take a look at, OK, what is the keyboard interaction for a particular widget? And within the acceptance criteria of the story, we were describing that. Left and right arrow would expand the accordion. Once it's expanded, you can use your up and down arrow keys or your Tab key to then continue down and move through that interface.

And so again, we're basically dealing with the same concept of looking at focus order in keyboarding, but breaking it down to widget level, so that I can then use this wherever I use an accordion-- for example, an accordion interface. I can pull up this example in my archive and be ready to use it again somewhere else that we're using this particular widget.

So again, with this diagram, I've included the elements of the acceptance criteria from the user story to describe what should be going on, and so again, that the engineer knows what he needs to do, and the tester knows what kinds of tests that need to be written to validate this functionality. So the relevant ARIA design patterns for this particular widget would be the accordion and the checkbox.

So one of the things that I always want to do when I'm creating these specs is basically to account for, what is an assistive technology going to need to be able to work appropriately for a user? So every element needs a label. Every element, we need to know and express what kind of role that element has, whether it be a checkbox or a radio button or an input field, and what sort of state is it in.

In the case of the accordion, which is why I used this example, for example, is it expanded or

not? Can I get to the elements inside of it? And basically, the label, role, and state for any element, when focus is put on it, should be announced immediately to assistive technologies. And there can be additional ways that we can associate things like label, role, and state with an element, even if it's not a visual label, including ARIA attributes like aria-label and aria-labelledby.

We might want to also include additional information to be announced to the user. And I talked about that content behind the content or screen-reader-only content. And within an iOS or mobile context, often, we hear it described as a hint. Within a web context, it might be described as a description. And often, it is the ARIA attribute aria-describedby that's used to actually assign that content to an element. And what occurs is when an assistive technology, a particular screen reader encounters that element, first they'll hear the information about its label, role, and state. There will be a pause, and then you'll hear the additional description.

And the thing is that when we think about writing this content for any element, the things that people absolutely need to know, like label, role, and state, or need to be heard right away, should be included in one of those values, because you can essentially, through your preferences, turn off the verbosity of a description or a hint being announced. So it's really intended for additional information that may be helpful and provide some additional context that isn't necessary to actually achieve what we want them to achieve.

And ultimately, what happens is that I want to basically notate, what should I expect to hear when a screen reader is reading back or announcing the content? And so I'll usually include an approximation of what would be announced by an assistive technology to give the engineers and the testers an idea of what they hear so that they know what to validate against.

So then, here's an example of my actually documenting information about a particular widget. In this case, it's for the neighborhood accordion that we've been looking at. And so clearly, I'm just saying that the label is "Neighborhood." The actual parent element of the accordion should have the role of a button. I should be defining what its state is, either expanded true or false.

Is there an additional description that I want to communicate to users? In this case, the description being "Select a filter to narrow your search results." And while it's useful, it's not critical for them to be able to use it, but it helps to give context to what this thing is. And then, finally, I'm giving an approximation of what elements would be read back to an assistive

technology and in what order, so that then, an engineer and a tester can validate that.

And then, finally, an example of how I would include that in my design spec. And using the notation element that I had shown earlier for communicating information from assistive technology would find important. So labeling, including its label, role, and then aria-expanded, which would be an example of an additional property or state.

And then finally, what would be expected to be announced by an assistive technology. So a diagram like this can very easily be utilized by an engineer to then build this functionality, and a tester can then easily write test cases against it. They basically have all the bits and pieces that they need to be able to both implement this functionality and test against it.

OK, and then finally, we're going to take a look at content structure. And I talked a little bit about ARIA landmarks, and we'll take a look at that in a moment. So first of all, by default, we should be using the semantic structures available in HTML, whether it be tags like the main tag or the navigation tag.

We should be using elements like heading levels to define informational hierarchy. Other container elements, like the fieldset and legend tag, to group similar or related groups of elements, let's say, within a form. Using structures like unordered or ordered lists, and then defining different regions. Some of the regions are defined now through the tags like main or nav that are part of the HTML5 spec. Or we can actually create what I would consider a generic region and then label that to give the customer an indication of what this area is. An example I used earlier for a mail application would be creating a generic region, and then giving it the label Inbox.

OK, so let's take a look at an example of how we would then mark that up. So again, we're looking at this search and filtering interface. And I've basically just easily drawn rectangles or outlines around some around the different groupings of elements within the interface. So we talked about the filters on the left, the main content of the search results, the toolbar for reading and sorting alphabetically, and then the pagination controls. And on the right, I'm documenting, what kind of role does this region have, and then a label that would be communicated to assistive technologies.

And this information shows up in a couple of different ways. An example within a screen reader would be that they can bring up a list of these landmark regions to see how has the content of this page or application been broken down and what regions are information

grouped into, so that they can actually use that list as a navigational tool.

If focus is currently on the filters on the left, I can read the list of landmarks and quickly jump down to the pagination controls at the bottom. And so it serves a number of different purposes, to be able to help to clearly define group elements within the interface.

And then finally, we're going to take a look at our last method here of documenting accessibility requirements. And that would be through test cases. And so basically, we've been moving through how things would occur within the software development lifecycle, from originally writing the user stories to help define the functionality and its acceptance criteria, those user stories then being picked up by designers to help actually flesh out and describe visually what the user experience is going to be, and then finally, what a test case might look like to validate these requirements.

So at a high level, this is a bit of an eye chart. For those folks that may have some familiarity with the Web Content Accessibility Guidelines, you will recognize these as particular guidelines. These are examples of ones that will be taken into consideration at a more macro level for a particular page or screen within an interface.

The tester would be looking at concepts like, does a particular element communicate the proper information about itself and its relationship to other elements in the interface? Or meaningful sequence. Well, we've, actually been dealing with meaningful sequence at a high level in that first diagram, where we looked at tab order. And so just an example of some of the different guidelines that are important initially to look at, really, for any interface that you're defining. And then we'll also take a look at things that we want to articulate at a more individual component level.

So here at the component level, we can look at things as either a single component, like an image or a form element like a button, or we may have complex components, like one that we've been looking at all along here, which would be an accordion, which is actually a composition of different individual single components.

And so as a tester, I may write a test that is all about validating the functionality of a checkbox. And really, I just need to write that test once and use it again and again. Or, I may include that as a sub-element to a test that I'm writing to validate the functionality of a more complex component, like an accordion. So one of the ways that test cases are often written can be broken down into a format of Given, When, Then.

And I take a little bit of inspiration from a presentation that I saw at the CSUN Assistive Technology Conference that occurs every year by a woman named Sarah Pulis. She gave a presentation called "Reusable Acceptance Criteria and Test Cases for Accessibility." And again, if you download the deck, you can find a link in the speaker notes or within the Word document to her presentation, which is available online today.

So the format that we're going to be looking at today is referred to as Given, When, Then. And so given some initial context that we expect to be true, or it may be describing the state that an application is in at the time-- so given that I am on the login screen, let's say, for an application. When some action is carried out, or an event occurs-- so maybe I'm validating what happens when someone clicks the Submit button when they go to submit that login form, is there an error to describe, or do we just then forward them onto the next screen? What happens then?

So then a particular set of observable consequences result. And I may actually write this format again and again and again to create various tests that validate a single user story, including those that address accessibility requirements. So we're going to take a look at an example of how that would show up.

So basically, to address acceptance criteria for an accordion, I would write, Given that I have focus on the heading of an accordion, when I press the Enter or Space key to toggle the accordion, then the associated panel toggles between expanded or collapsed. And here I am including within the test case the Web Content Accessibility Guidelines criteria. And this is not an exclusive list, but an example of here are some of the guidelines that I'm actually saying should be validated as part of this test case.

And then what kinds of checks as a tester am I going to be looking at and including as part of this particular test case? So I'd be looking at, well, is that the parent element of that accordion conveying its role as a button? Does it have a name? Is there some sort of programmatically associated label? And on and on and on, I would be validating these different elements so that I can ensure that when someone who's either using a keyboard or an assistive technology is getting the right information and the control or that the UI element is working as expected.

OK, so we are drawing to a close here. We've got one more thing to look at in terms of the-- let's see here. So essentially, what I've put together now-- we've introduced these three

different methods to be able to document and understand and express accessibility requirements. And I've basically packaged these things together in what I refer to as a Digital Accessibility Toolkit.

And so on the first page of the presentation, I had referenced a bit.ly link, which will take you to a Dropbox folder where you're going to find these elements. You're going to find the user story example, the one that was included here in the presentation. You're going to find different ways that you can download and open up that set of accessibility annotations. So there's an Illustrator file version, and there's a SVG file. There's a lot of different graphics programs that you can import that SVG file into and start working with right away.

I've also included an Adobe XD Creative Cloud wireframe example, as well as exported PNG image format versions of all of those screens, so depending on what software you have yourself, you'll still be able to open and see those examples, as well as in the presentation. And then this test case example as well. And so I've tried to provide all these assets in formats that can easily be opened, regardless of whether or not you have a variety of different kinds of, let's say, art software.

So additional considerations that you might also document within your design specs for your user stories might include things like, is there a style guide that needs to be referenced? Are there different design pattern libraries that are already defined or should be referenced as additional context for a story?

Are you looking or validating whether or not there is proper color contrast within your UI? You may actually have a diagram where you're saying, OK, this text against this background has this particular contrast. We've validated that and, so the tester can validate that as well. And so when the final product is in, let's say, a development or staging environment, they know what the contrast should be, and they can validate against that. Other additional tool tips that need to be expressed within the interface or keyboard shortcuts.

We didn't touch so much here on touch and gestures. But is there something unique about this interface when it shows up within a mobile context? Are there combinations of touch gestures that are relevant only for that particular platform? Are you looking at the interface and evaluating what it's going to look like when text is resized? And then, additional content for assistive technology users, you might express using either the aria-describedby attribute within a web context or additional hints within iOS applications, or what we call a content description,

which is similar to a hint within Android applications.

So I've included some links to both resources that were used to put together this presentation, as well as ones that I think will be useful for you when sitting down and really trying to decide how you're going to document accessibility requirements within your particular project.

And then finally, I just want to say thank you. Very excited to be able to share the work that we've been doing here within Adobe. And we've began to actually continue to share that out with other designers and product managers, that the companies were actually seeing this approach being picked up by a number of different individuals and them really finding success with it. And so I hope the same will be true for you within your own work. And at this point, I'll hand it back over to the moderator. Thank you.

SOFIA LEIVA: Thank you, Jack, for such a wonderful presentation. We can get started with the Q&A, and I'd like to encourage everyone to keep typing your questions into the Q&A box.

So the first question that we have is, what are the pros and cons of expanding a filter or heading in the tab order on Space or Enter, versus leaving the tab order unchanged and relying on the arrow keys for navigation within a filter heading?

JACK NICOLAI: I'm sorry, could you repeat just the first part of that question for me again?

SOFIA LEIVA: What are the pros and cons of expanding a filter heading in the tab order on Space or Enter, versus leaving the tab order unchanged and relying on arrow keys for navigation within a filter heading?

JACK NICOLAI: Got it. OK. Well, so I think design patterns, even within the ARIA design specification, are essentially guidance. I think that if you're building an application that is a desktop application-- I mean, that's really what the ARIA specification aims to do, which is to provide guidance for people building web applications to mimic the functionality of a desktop application.

And I think that there can be some pros and cons in trying to do that. Web applications have become incredibly powerful and much more desktop-like than they used to. But I think users still have a familiarity and an expectation as to how keys like the Tab key should work in many contexts.

And so in this case, it's a stylistic choice. There are times where I may choose to also allow the Tab key to move through elements in the interface, because, let's say, the child element of

that accordion may-- because it is an interactive element, it is by default included in the tab order within the DOM. That's just the way a browser works by default. I would actually be overriding that functionality to only allow the arrow keys to move up and down within the contents.

Another example would be, let's say, a dropdown menu. I'm thinking of, actually, specific examples of stuff that we're doing on adobe.com, where in a dropdown menu, you may have a series of items in that menu. Well, within each item, each item may actually include some elements like links and buttons. And so, if not allowing the affordance of the Tab key to be able to get to those elements, what would be the best way to provide that access?

And so there are combinations of patterns you can work with to do that. But at times, I'll basically try to remove some of the barriers that some of the design patterns can actually cause to actually get to a particular element. And without getting too much into the detail of those, but it is to say-- some of it is stylistic.

I think in terms of pros and cons, I think it all comes down, as well, to user testing. Whatever your ideas are about how a user should be able to use an interface, you've got to eventually get that in front of them and validate whether or not that makes sense to them. And so that's where, ultimately, the pro and con is going to get answered is, it's a pro if your customers actually find it easy to use.

SOFIA LEIVA: Great. Thank you. The next question we have is about the Make Reservation link on the web page layout example. What's more appropriate text to use for buttons to make each unique so that screen reader users doesn't [INAUDIBLE] here make a reservation out of context?

JACK NICOLAI: Yeah. That's a good one. And I actually am glad you brought it up, because the way that this interface is created, it's intended to bring up a question like this. Because no matter how hard you try, you're going to work with designers or even product managers that are perfectly OK with this kind of interface, where you've got a number of calls to action-- in this case, links or buttons-- that use the exact same label, which from an accessibility perspective really isn't ideal.

And the reason for that is, if I'm using a screen reader, one of the affordances that I have to be able to bring up a list of all the links that are available within that particular screen. And if I have 100 search results, or even 10, really, that all simply just say Make Reservation, that doesn't really give me the context I need to be able to choose one of those buttons versus

another.

And so there are some techniques that can be used to provide additional context. If you look at one of the cards for the restaurant itself, the element that would be primarily unique for each individual card would be the restaurant name. And so I might use the aria-labelledby attribute to basically compose a name that would include-- it might be read off to an assistive technology user as Make Reservation and then the restaurant name. So Make Reservation, Chili's. Make Reservation, Macaroni Grill.

And so I'm utilizing the content that's there in the page already and using ARIA to be able to construct a name so when that list of links gets brought up, it would essentially have the additional content associated with each link content to be able to differentiate each of them. Now, this is just one example. Depending upon your interface, you may or may not have content you can use to help differentiate it. And it's a case-by-case basis, where you'll need to think about the best way to approach that problem.

SOFIA LEIVA: Thank you. The next question we have is, it sounds like there's a lot more work for designers. How does one adjust for the time-slash-effort needed during the design phase?

JACK NICOLAI: Yeah. I've spent a lot of time, at least within this presentation, on the design piece because I think more that it is a gap right now. It's an activity that a lot of designers are not engaged in.

And I think that what's going to occur, and what I've seen occur, is there's an initial investment upfront, both where product managers and designers are, number one, getting educated about how to be thinking about, how to be writing about accessibility requirements. But after a while, it becomes like anything else.

I mean, it's not to say that this additional markup and design doesn't necessarily take additional time. But where you're going to over time find savings is where I was showing examples where we had the whole screen that we'd marked up versus components. Because typically, you're designing a design system, you're designing a set of UI elements that you're going to use again and again and again, especially if you're dealing with a web application or a corporate website or that sort of thing.

And so you go through the activity of defining a lot of these things once, and then you can basically just pull them into new designs that you're constructing with those same elements, so that it takes less and less time as you go along. And really, I think it has to come down, as well,

to an agreement and understanding with your product team and your product managers that this is work that needs to get done.

I mean, ultimately, if you were to ask the question, who shouldn't be able to use your website or your product, you would probably say, well, there isn't anybody that shouldn't be able to use our product or website. And so the question then becomes, well, do we take the time to define what is needed to be able to make our website work for someone who can only use a keyboard?

Well, ultimately, the answer to that should be yes. And I know not every organization is at the same place. But I think that there are ways that individual roles can push back up towards the product team to say, this design spec is not done until I've done these activities. It's not going to come for review, or I'm not going to hand it off to an engineer, because it's missing stuff.

It's like, if you submit a design, and none of the buckets had labels on them, like Make Reservation was just empty in this interface. That would not pass. That would not get handed off. And so I think at some point, we have to just say, look, these are the activities that have to get done within a design exercise. Otherwise, the design itself is not complete.

SOFIA LEIVA: Thank you, Jack. The next question we have is, does your QA team have AT users that do the test, or folks who are familiar with how to use AT, or both?

JACK NICOLAI: In some cases, yes. In some cases, no. And we have a lot of products at Adobe, and developed in multiple countries. In my own portfolio, I have half of my teams are in India. And in some cases, we have individuals that are native users to different types of assistive technology. In other cases, we are applying test automation. There's a few common libraries out there to test against the Web Content Accessibility Guidelines, and so certain things will validate through automation earlier in the process, and then bring either users in or internal testers in later to validate.

And in some cases, it may be that we need to reach one of our accessibility specialist vendors that are out in the industry to help either do the testing themselves, because they'll often employ people that are native assistive technology users-- but it really depends on the team and what resourcing they have available. We'll come up with a range of solutions to help them cover all of the testing.

SOFIA LEIVA: Thank you. The next question we have is, what is your work flow between accessibility

specialists and designers?

JACK NICOLAI: Well, I do a lot of consulting. Typically, my best-case scenario is, I'm initially sitting down with the team that's going to be-- and if you're familiar with Scrum or Agile methodologies, when you are sitting down and actually considering to take up new work, the assumption, typically, is that a product manager has written the user story to describe a certain set of functionality, and then everybody sits down to talk about what this feature is so that they can ask questions and flesh out, if there's something that they don't understand, or they see as missing, that basically, that user story gets massaged into a feature description that everybody understands.

And then from that point, I will often sit down with designers as they're beginning to wireframe out an interface. And I ask them questions. Well, so how does the user get from point A to point B? When the user puts focus on a particular element, what gets announced to an assistive technology? And often, those questions-- for an inexperienced designer in terms of accessibility requirements, this is just not a part of their thought process right now.

And so it's both the activity to educate them and the activity to uncover and make sure that these things are being thought of and addressed, and then those things are being integrated in ways that I've illustrated here, that they're being illustrated, then, within their design specs.

And from that point, then, often my next touchpoint is the engineer. And so we've basically had my influence on the user story. I've had my influence on the design specs. And then when that handoff occurs to the engineer, there are technical things that they're going to have to implement that aren't really the designer's responsibility, and they aren't the product manager's responsibility, but are still things that-- I'll often then do technical consulting from an accessibility standpoint.

SOFIA LEIVA: Great. Thank you so much. I believe we have time for my question. Are there any trends you are seeing for how to make things easier for designers to produce accessible products that [INAUDIBLE] have to point out?

JACK NICOLAI: Any trends. Well, I think we're starting to try to start one with this kind of documentation, really. Because while I was seeing bits and pieces of this kind of approach bubbling up at different companies and with different accessibility professionals and designers, now we see a lot of different designers that were like-- I'm super passionate about accessibility, and we're trying to figure out how to express this within our work.

And so right now, the trend I feel like, at least, is growing is this both desire and seeing these requirements starting to show up within design specs from a software perspective. That's something that we are really talking about and considering, how do we provide software-based solutions, like tooling, essentially, within our products that will make this process easier?

Now, in particular, products like Adobe's XD, which is for wireframe and design prototyping, as well as other products like Sketch. Now we are thinking about how we can provide tooling within those products to make this process easier. In the meantime, it's why I put this toolkit together, so that you can just go download these assets and start to pull them immediately into the design work that you're doing and find some value. And so, I guess, keep your eyes open in terms of Adobe XD in the future.

SOFIA LEIVA: Great. Thank you so much, Jack.