LILY BOND: Welcome, everyone, to the Future of Video Player Accessibility. My name is Lily. I'm joined today by Matt Schweitz, the engineering manager at YouTube; Vlad Vuskovic, the product manager at YouTube; Eric Boyd, the director of product at JW Player; Steve Heffernan, author of Video.js; Terrill Thompson, the technology accessibility specialist at University of Washington; and by Greg Kraus, who will be moderating the webinar. He's the IT accessibility coordinator at North Carolina State University. And with that, I will hand it off to Greg.

GREG KRAUS: Thanks, Lily. Welcome, everybody. My name is Greg Kraus. I'm at North Carolina State University, where I'm the university IT accessibility coordinator. And for today's session, where we're going to be talking about web video accessibility, I just wanted to give a little introduction first for the topic. So I'm going to give a brief introduction, and then we're going to hear from the four vendors that we have with us, from Video.js, YouTube, JW Player, and Able Player. And then we're going to have some time for Q & A at the end. So if you have questions, please post them in the Questions area, and we can get to those.

So just to lay some groundwork, so what do we mean by video player accessibility? So a lot of times, we think about video accessibility, we think captions. And yes, that is part of video player accessibility, but we're taking up a little larger look at video player accessibility and looking at the actual player itself today, and how these particular vendors are approaching accessibility. So asking questions like, does the player require using a mouse? Could you use it with a keyboard? Could you control the video player just using voice recognition software?

Can it support audio descriptions, or what's sometimes called described video? Could it support something like a separate sign language track? Do the player controls themselves have enough contrast for users to be able to tell the state of the buttons? Is the button enabled? Or being able to get the information they need there.

And can the player itself be customized to meet the user's needs? If they need maybe even more contrast than the player provides, does it allow those types of functionalities? So that's what we're really looking at today, with the video player accessibility.

So one thing I wanted to say that this is not an endorsement of any one particular product. And the people we have represented on the panel, this is not an exhaustive list of everyone who's working on accessibility and everyone who's made accessible video players. What this is, it's just a conversation with several developers in the field. And we have several people

representing some video players that are in quite high usage around the web, so I think this could be a great conversation.

But we did want to give acknowledgement to some of the people that are doing other work with video player accessibility, that we couldn't necessarily represent everybody on the panel. And even this is not an exhaustive list, but just to mention a few, Kaltura has done good work. Mediasite has done some.

BBC iPlayer-- it's not really a player you can use for your own stuff, but the BBC content can be viewed through there. PayPal has developed a video player that's accessible, OzPlayer, which is a little newer one in the field, Acorn Player, and there are several others, too. So don't take what we're saying here as you have to use these players. We just wanted to give acknowledgement to some that are out there.

So what we hope to accomplish today is we want to have a conversation about how the developers are approaching accessibility, what challenges they are facing, and what direction they're hoping to go in the future. And so with that, I'm going to turn it over to Steve now from Video.js.

STEVE HEFFERNAN: Yeah, hi, everyone. My name's Steve Heffernan. I was the original author of Video.js and continue to be the lead of the project. Video.js was originally created just to add controls to the HTML5 video element. But from then, it's grown a lot, including adding support for Flash video, other players like YouTube and Vimeo, and then a lot of great features on top of that, like playlists, analytics, and even virtual reality. So the project has grown a lot.

On the next slide, you can see an image of the player, in case you might recognize it. But also, a lot of companies will customize the player a lot, so you may not even realize you're using it when you are. On the next slide, you can see a handful of the customers using Video.js in some way or another, including Twitter, Tumblr, Instagram, Dropbox, United Airlines, Toyota.

And then the company I work for, Brightcove, is an online video technology provider, and we're working on building the next Brightcove Player on Video.js. And Brightcove powers sites like NBC, AMC, The Weather Channel, Showtime, lots of high-end video property. So lots of great sites using Video.js. It's being viewed billions of times a month. And so improvements in Video.js can have a big impact on accessibility across the web.

On the next slide, probably the most important thing to understand about Video.js is that it is a

completely free, community-built project. No one is making money off of Video.js disk itself. And all of the features, including accessibility features, have been added by contributors who have stepped up and said, I want to improve the player in this way. So if you are at all interested in getting involved in helping push the player forward and impacting video on all of those sites I mentioned, please do.

So on the next slide, some of the experts that have contributed so far include Greg Kraus, the moderator. Owen Edwards of YouDescribe.org has been helping us navigate the WCAG 2.0 standard. And then Karl Groves, now at Tenon.io, helped us improve our HTML.

So I wanted to talk, just kind of give a high-level overview of some of the features we do support. So we've supported WebVTT captions from very early on, even before browsers were supporting WebVTT captions-- and again, because a contributor stepped up and said, I'm going to implement this in the player. But since then, captions have got a lot more complicated. Browsers have got better at supporting them.

And today we need to support advanced features, like vertical languages in captions, and multiple captions tracks running at the same time. And none of those can overlap, and so you end up building a collision detection engine, almost like you're building a video game, just to display captions. And so what we ended up doing is partnering with Mozilla, the company behind the Firefox web browser, and helping them build VTT.js. And now both Video.js and Firefox share the same code base for displaying captions. So that's been a great project, and it's added a lot of great captioning features to Video.js.

Then we also support speech and tab navigation, which has kind of come from essentially just always using HTML as our controls. So even when we're playing back video through Flash, we've always used HTML controls over top of the Flash video. And so that's made it easier to support tab navigation. It's made it so we've only had to build accessibility once, and we get to take advantage of a lot of the browser features for accessibilities. It doesn't get you all the way there. We have had to do some additional work. But it gets you pretty far.

So then I wanted to just give you some insight in some of the challenges that we run into while building a player and providing accessibility. Some of the things that have come up in-- when we tried to make the player more customizable, it resulted in poor accessibility. The first example, divs versus buttons, on the next slide, you basically have an option of either using the button HTML tag to create your button or using the div HTML tag, which is kind of like an

empty HTML tag, and then adding your own button functionality to it and the ARIA role of button to it.

So you might be thinking, well, why the heck wouldn't you just use button? Well, the problem is that browsers and other web developer tools often add their own styles and functionality to the button element. If you can see the image at the bottom of the slide, I've taken Video.js and dropped it into the popular Foundation framework for building websites and then switched the divs to buttons. And you can see that the buttons then turn bright blue, they get bigger, they break out of the control bar and just completely mess up the design. And so in order to switch to buttons, we need to do a lot more work around protecting our own styles so that the player can continue to be as portable. But we will be doing that. We will be switching to button, because the accessibility experts keep saying that that is the right way to go for supporting the long tail of screen readers.

On the next slide, we tried to make our control bar extremely flexible, so anyone can just add a button to it and it would just automatically flow into place. However, when adding that flexibility, we also kind of messed up the tab order. So when you tab from-- like you tab into the player, you hit the Play button, and then from there, it goes all over to the right-hand side of the player to the Full screen button, and then starts going to the left-- to the volume, to the Closed Caption button. And so it's just generally messed up. And so in the next version of Video.js, 5.0, we're going to be using a newer browser technology that will allow us to keep the flexibility but also maintain the correct tab order.

So, yeah, that's kind of an overview of things. We're going to be switching the buttons, fixing the tab order. We're going to improve the frame styles so it obviously at a specific button. We're exploring audio descriptions with YouDescribe.org, as well as WCAG 2.0 compliance and everything we need to add to Video.js in order to support that.

So yeah, we're generally very excited about the future of accessibility in video players. And as I said, if you're interested in getting involved, on the next slide, you can always reach out to me. On Twitter, I'm @heff, H-E-F-F. Or the project lives on GitHub at github.com/videojs-- no space-- /video.js. And you can go there, submit an issue, and we can start a conversation. And that's it for me. Thank you very much

GREG KRAUS: Thanks, Steve. And as people are presenting, go ahead and keep asking questions. Several are coming in. And we'll gather those up at the end for our Q&A session. And so now I'm going

to turn it over to Matt and Vlad from Google.

MATTHEW SCHWEITZ: Hi, everyone. I'm Matt Schweitz. I'm an engineering manager at YouTube for the Captions and Accessibility team. I'm here with Vlad Vuskovic, who's a product manager on globalization and handles a lot of our captioning product, as well. I'm going to talk a little bit about how we approach accessibility in general at YouTube, if you'd go to the next slide. And then we'll briefly talk about what we've been up to, what we're doing. And then we'll cover a little bit about what we plan on, what's next for YouTube.

So for approaching accessibility, especially for a large organization, it seems like it should be pretty straightforward. First, get buy-in from the org. Next slide. And then step two would be-- that's it. There is no step two. Once you have buy-in from the org, everything else, you would think, would follow.

The next slide will show the mission for Google and YouTube. And you can see, if you go to the next slide, as well, we highlight things like being universally accessible, empowering the world. So given this, you would think it's a no-brainer. There's nothing else you have to do. The organization has said that these are the priorities, so everything else will follow.

In essence, it doesn't really work this way. There's always something that seems to trump accessibility in a large organization. Priorities don't necessarily align in favor of accessibility. So I'm going to talk a little bit about how we've shifted, specifically at YouTube, to account for accessibility in our product. And this covers the player and embed and on-site contexts, as well as our website and mobile apps in general.

So what we found is that thinking about accessibility as a dimension of overall product quality has been very effective. So what does that mean? It means that you need to actually consider all aspects of product quality that you can test, measure, and integrate, and accessibility folds into that and becomes, in essence, a priority along with other aspects of the product quality in general. So things like UI polish, things like bugs that may surface to only certain types of users, test coverage-- these are all aspects of product quality, and accessibility starts to bubble up as a main aspect of that. And once you have attention on the general aspects of product quality, we found that actually paying attention to accessibility becomes more of a priority.

So the next slide will tell a little bit about how we do this in practice. As far as testing, a lot of it has to do with QA, and that can be developer QA, official QA from QA testers. But it also has a

lot to do with education and evangelization within the organization. So making sure developers are aware of that product quality and how accessibility plays into product quality, educating them with training, making sure that accessibility and overall quality is accounted for in design.

And then tracking progress-- this is something that we found to be incredibly effective on the accessibility and overall product quality front. Tracking bug counts, exposing status of different teams on a dashboard. And you can see these graphics sort of show little spark bar charts of sort of team product quality above burn down chart-- this is specifically for accessibility. And allow teams to really track their progress.

The last thing is putting accessibility and product quality on the critical path to release. So blocking launches of new features until accessibility and product quality is accounted for in a test plan. Making sure teams are thinking about that before they launch, as opposed to reacting to it afterwards.

So I'm going to talk quickly about things that we've done, especially recently, in this context. In practice, really we think the basics get you pretty far. So things like semantic markup, sensible DOM, attention to contrast, in terms of icons and colors, and then paying attention to tab indexes and roles. Specifically, some things we've learned in the player, and in general on YouTube.com, the website, handcrafting tab indexes it is better than trying to deal with them automatically.

We had spent a fair amount of effort trying to increment tab indexes automatically with JavaScript, allowing them to dynamically react to new features. And in the end, it just turned into a complicated mess. We found that same DOM with tab indexes that you manually add and account for, trying not to be too fancy, gets you pretty far. Also, we found that shared tab indexes work very well. So having a DOM element that shares a tab index and allowing the index to flow through them naturally works pretty well.

We've started to use aria-owns quite a bit in the player, especially in the settings and other dialogues. And that has been very useful. And then the last thing is that using labels are cheap. And you can't really overuse labels. The Full screen button is a good example. It's kind of a tricky thing to get right, in terms of where focus goes when you go to full screen mode. And we've decided that focus will sort of bubble up to the player, the main player element itself, using a negative index. But it has a label so you can know where you are, and you won't get confused around play/pause, full screen toggling.

I wanted to talk a little bit real quick, about some recent improvements we have made in the last year. Tab ordering in the player has been significantly improved and is now quicker to find. The seek slider, the slider bar to scrub through videos, has been made a lot more accessible. It was not previously possible to use with keyboarder screen readers. And then the Settings menu and Share panel have also been made more accessible. And you can actually now skip ads with keyboard and screen readers, which is pretty useful.

And then lastly, we've added a global label to announce new things. This has been particularly useful for dynamic pages and things that kind of flip in and out of the UI. So we have sort of a general div that will alert on new things, like that an ad is playing, or that if you're in an autoplay context, the next video is coming up when a countdown starts. And this has been very helpful to deal with browser quirks, especially for IE and how it deals with role alert surfacing.

So next slide is going to talk a little bit about the player and captioning.

VLADIMIR VUSKOVIC: OK, so to make captions really great, we need to do two things. First, we need to ensure that the viewer experience is really great. And second, we need to make sure that as many videos as possible are captioned. So next slide.

To ensure the quality of user experience is really great. And [INAUDIBLE] captions is a core part of the player. We also have preferences that allow users to select when they want the captions to show up, and also that allows them to customize how the caption should show up across all of our platforms.

In order to ensure-- next slide-- that we have a broad coverage of [INAUDIBLE] caption, we're rely on three strategies. So in the first place, we offer creators a series of tools that enables them to upload captions. These captions tend to be the highest-quality ones, but they account for only kind of the tip of the iceberg. Basically only a small subset of the videos are captioned that way.

In order to scale and cover a lot more videos with manually-crafted, high-quality captions, we are relying on the crowdsourced platform we've launched recently. And that allows, basically, fans to generate captions for the content they love. In order to scale even further and caption the lower end of the videos, we rely on autogenerate captions. These are not perfect, but we are constantly working on improving them. So, next slide.

So I'm really excited about this part of the project, which is the fan subtitles. This UI quickly shows how the users can contribute and submit captions for the content creators to review and approve. And we are really convinced this is going to help us really scale both the quantity and the quality of captions across YouTube. So, next one.

This is the last-- the last slide is basically what we're trying to work on next. So we plan to-- currently the fan subtitles is in a beta, and we plan to have a wide release of the product. We're going to be working on more refinements on accessibility in general and on discoverability of keyboard shortcuts. We'll put more attention on mobile. And we are going to invest in ASR improvements for our caps. The last thing, we are going to also be working with surfacing audio descriptions.

MATTHEW SCHWEITZ: So thanks everyone. If you do have feedback, please use the Send Your Feedback link at the bottom of the YouTube.com desktop website. You can tag it with a11y or accessibility. People do review these things and escalate them to our teams.

VLADIMIR VUSKOVIC: Thanks.

GREG KRAUS: All right, thanks, Matt and Vlad. We're going to go ahead and turn it over now to Eric Boyd from JW Player.

ERIC BOYD: Great. Thanks, Greg. Also wanted to have a quick thanks to Lily at 3Play for organizing this. And just to reiterate what you had said, Greg, when you started this off, this definitely isn't an endorsement. And I think it's important to mention that a lot of us actually do work cross-video player products to contribute to accessibility standards.

So just wanted to give a quick overview of JW Player, in case you're not familiar. Our product vision is to provide the best possible viewing experience across all devices-- desktop, mobile, and even connected TVs. We work very closely with developers, we do user experience studies, and we try our best to keep up with the ever-changing market needs and contributing to these evolving standards. Next slide, please?

So we were founded in 2008 as actually one of the very first video players on the internet. So we have some strong roots in Flash. Actually, YouTube started off that product using JW Player. Kind of a fun little fact. Right now, we have about 95 employees and about 17 billion videos are watched through JW Player on a monthly basis across about 2 and 1/2 million

domains around the world. So just about 50% of these publishers are media or over-the-top broadcasters. 25% of them are creative agencies, development studios. And then we have a lot of advertising network and syndication networks as well. Next slide?

So we talked a lot about-- I know Steve talked a lot about tab control, buttons versus divs. Since JW Player is live on so many different websites around the world, there's a lot of use cases. We are a Flash Player and an HTML5 player, and the player intelligently chooses which one of these rendering modes to do. So that did actually have a challenge of two separate video controls-- there's Flash controls, and then there's HTML controls. And when you would try to tab through those elements, you would kind of get lost in a tab index.

So what we decided to do, we decided to try something a little bit new, where we created a tab-in/tab-out model for accessing our video player. Once you have focus on the video player, you get access to a set of keyboard controls that are keyboard shortcuts, that you can quickly access the main, core video playback needs. You can play, pause. You can seek. You can adjust volume. You can go into full screen. Some issues with this that we know is that these elements are not actually exposed to a screen reader. So that's something that we definitely want to look into. Next slide, please?

So multiple audio renditions are actually really important for accessibility. Unfortunately right now, this is only really supported in two to three types of streaming formats. It's not actually built into or supported by most of the browsers in the regular HTML5. But we do support multiple audio renditions-- so the ability to have a descriptions audio track. And this is through HLS. And if you're familiar with MPEG DASH that's also supported in that streaming format.

But this functionality gives the viewer an option to switch between those tracks, and then the audio will switch back on. I believe Netflix just announced that they were going to start doing this with their video content. And we provide an API for publishers to access these audio renditions, and they can provide some special UI on the page, if that's necessary. Our player will actually also automatically detect which language, or if there's a default on that audio track with descriptions to be played, it'll pick that to play first.

And we've talked about closed captions. JW Player will backfill support, similar to what Steve was talking about. Where captions weren't supported in the past, we would add that functionality, so that you would universally get those caption support anywhere the video is being watched. You can also, like, transcripts, and using our API, you can do a pretty simple

demo of getting transcripts visible to viewers. And then those transcripts and captions can be styled however the viewer actually requires them to be styled. Some things that we're working on filling in is adding positioning elements and supporting, like Steve said, vertical captions, as well. Next slide, please?

Yeah, so there's definitely a lot of challenges with accessibility, and especially being a third-party application that does get sort of inserted into other publishers' websites, where we don't have control of the tab Index, or where the video player is going to be placed. And we also need to support the latest browsers, and then have to support the standards that weren't actually available yet in older used browsers. Something that we're hoping to accomplish and hoping to add to is getting actual video-specific interaction standards built into perhaps even the video tag, so it would be a fundamental part of the video player. Some other sort of just challenges is always trying to schedule when this work can get done, especially when there's, all of a sudden, some government mandate to add some new functionality to a video player.

So my next slide is just a summary. Oh, no, I have a road map. But I think I talked about all these with each of the earlier slides. We want to add support for 708. We want to make it easier to skin the player with CSS, so that you would get the same elements across both Flash and HTML5. So moving away from Flash and going to more of the HTML5 standards.

And then you can put the summary slide up. But feedback is always welcome. We have a very large, vibrant community of people who add things to our player. Always feel free to reach out to support@jwplayer, or even hit us up at Twitter at @jwplayer.

GREG KRAUS: All right. Thank you, Eric. So now we're going to turn it over to Terrill for our last presentation on one of the video players.

TERRILL THOMPSON: Thanks, Greg. I'm Terrill Thompson. I'm with the University of Washington. And first of all, I just want to say that I'm really humbled to be included on this panel. Just having Able Player included in the conversation with these other players is really an honor.

This started out as a personal project. I actually created an audio player for distributing music a couple of years ago and was just excited about the possibilities that HTML5 presented. And so using the HTML5 media API, built some buttons to control the player, and put some effort into ensuring that those were screen reader-accessible and keyboard-accessible, and controllable with voice, and that it worked well in Windows High Contrast mode, and so forth.

And having done that, by day, I'm actually technology accessibility specialist at the University of Washington, and I work for a project called DO-IT. which is-- it stands for Disabilities, Opportunities, Internetworking, and Technology. And actually, a little too quick on the draw on the slide advance. There we go. Thanks, Lily.

But so DO-IT has been around since 1993 and working on a variety of different projects, all focused on empowering individuals with disabilities to pursue challenging academic paths and careers. And with funding from the National Science Foundation and US Department of Education and various other sources, we have produced over 50 videos over the years, all related to kind of our mission and our focus. And they've always been captioned, they've always been audio-described, and we're always looking for new, innovative ways to distribute those videos.

And so we, a couple years ago, took the audio player that I had developed and started working on making a video player out of that, and really putting a lot of emphasis into demonstrating how incredibly cool accessible video can be. So that was really a focus, was building the demonstration player that would take all the things that were possible in HTML5, which include new support for audio description and support for chapters, and of course, support for captions and subtitles, and just really push that to the limit, and build a player using the principles of universal design that would really help us to sort of evangelize the need for captions and subtitles and audio description. So we had a player that we could just show people what is possible if you include these accessibility features.

And so we built that, and it's available up on the DO-IT website. And we gave a couple of presentations at conferences, like the CSUN Conference on Technology and Disability, and Accessing Higher Ground in Colorado, and got a lot of people interested. And so last year, we went open source with it, put it up on GitHub. It's freely available, Has a Wide Open License. And in particular, Ken Petrie of the Ohio State University stepped forward, willing to help out, and has been a partner on this ever since.

And the CIC, the Committee on Institutional Cooperation, whose logo appears here on the first slide, has been a real key player in helping to support the development of the player. So they have actually funded some JavaScript developers who have really put a lot of effort into this and taken it to the next level. So it is up on GitHub. It's at ableplayer.github.com/ableplayer. And we encourage people to check it out and to help with the development. So on to the next slide now, Lily?

So this is a screenshot of what it looks like on the DO-IT Video website, just an example. We have a lot of buttons, and those appear dynamically if certain features are available. So sometimes they're not available, but we encourage people to take advantage of all the features. The buttons, as I mentioned, are all fully accessible for screen reader users. They're fully accessible by keyboard and have good visual contrast, so you can tell when you tab to a button that it has focus. They're accessible by voice and have tool tips that appear by hovering or by keyboard focus, so that the voice commands are discoverable for somebody who's using speech input.

If we look at the buttons on the bottom row there, there's a caption button. Obviously we've got captions and subtitle support. There also is a D button for descriptions. We support WebVTT descriptions, which I believe is unique. There might be one other player that's doing that.

And it's kind of an interesting challenge, because what we're doing is exposing the text-based description at the appropriate time. So a VTT file is essentially like a caption file that contains description text rather than caption text. So we expose that text at the appropriate time in an ARIA live region, and so screen readers then announce that.

But it's tricky, because the person who scripts that VTT file has to really be good at finding the right place for that description to occur. And there's no way for us to tell, as developers, how quickly a person-- what speed a person's going to be operating at if they're using a screen reader. So we know when the description will start, but we don't necessarily know when it's going to stop. And so avoiding collisions with the program audio involves a little bit of guesswork.

So we've actually added a feature where there is pausing capability. So when description starts, the video pauses. So that's one work-around. But we found that that's a little bit clunky, in that if there's a lot of description, it's pausing frequently, and then the user has to replay frequently. And so that may or may not be something users want, and so that's included as a preference, and people can turn it on or off by default. And there's a Preferences button, which is really a key feature, in that everything we do, or a lot of things we do, are options for users to turn on or off.

So we also have support for adjustable playback rates. This is actually a really valuable feature. If people need to slow down the content in order to understand it, they can do that. Or if they need to speed up the content, as I often find that I want to do, as I don't have time to

watch a full video but I can just zip through it really quickly with those speed controls.

And there's also-- what's shown on the screen here is an interactive transcript. And that is assembled dynamically from the caption VTT file as well as the description VTT file. So it takes both and stylizes the description differently. And also, there's some hidden text in there that identifies that the screen reader users have audio description. And so that provides full access to both the description and the caption content. And as I say, it's interactive. So you can click anywhere in that transcript in order to launch the video at that point.

And its keyboard-accessible, so a person can tab through the transcript and press Enter on whatever text happens to have focus, and start the video at that point. Now that's another one of those features that would be pretty cumbersome if somebody was trying to tab beyond that onto something else on the page. So it's off by default. But again, in preferences, people can turn that on.

So let's go to the next slide, and just kind of a brief overview of how we do some of this stuff. We're using the HTML5 data attribute. So the ability to have custom attributes, like data-des-src-- that's the data described source attribute-- if we add that to any source element, then we can have a URL that points to the described version of a video. So that's an alternative to having WebVTT description, which we really still view as experimental. We can have a video that actually has the description extend, and then with the data describe source attribute, we link to that. And what happens is if a user's preferences say, I want audio description, then if the data describe source exists, then it'll actually swap that in for the source file. And so the user then gets the described version of the video.

We also have a data sign source. That's a new feature that we've just rolled out recently. And that will show a companion sign language video. So you actually have two videos side by side, one with a sign language interpreter, and both of those are controlled by the same set of player controls.

So and then I think I've talked about everything else in this list. So let's just move on to the final slide. And it's actually interesting hearing Steve talk about his struggles with the button element. And we're not in as widespread of use, obviously, as Video.js, but it'll be interesting to see if we run into some of the same problems.

We are using button, and we've tested button extensively. And just the URL of the test page--

tinyurl.com/button-a11y-- so you can kind of look at what we did. But since the button really is the key to the interface, most of what we're providing access to is via buttons. We really wanted to get that right. And so we looked at every combination of title attributes and aria-label attributes and hidden text nested inside the button, and glyphs for the visual look, and tested those with every combination of screen readers and browsers, and found that there was a particular combination that worked, and every other combination did not. Even accommodations that you'd think would work, they ran into problems, whether it was either screen readers redundantly announcing labels, or not announcing labels at all. And yeah, this is a problem that we run into all the time with ARIA and trying to meet the needs of users of different screen readers and different browsers.

And so since we just have this one user interface element, we really put a lot of effort into trying to get it right. And so we did test with JAWS in both Firefox and IE, with NVDA and Firefox, Window-Eyes in IE, VoiceOver in both Mac OS and iOS, and Talkback in Chrome, and found that there was a combination that actually works well in all of those combinations. So you can see that on our test page.

So that's all I have to say about Able Player.

GREG KRAUS: All right, thanks, Terrill. So we're going to go into a Q & A time now. And for the first question to the group, I want to pose this. The question is how much of this accessibility responsibility is the responsibility of the video player application that you've developed? Or is it the responsibility of the browser? And a related question that came in through the Q & A was asking about the issues of network security, and what if corporate network security blocks JavaScript, and all of a sudden, those controls are gone? So whose responsibility is it to make these video players accessible, and how do we deal with that as developers of these applications?

STEVE HEFFERNAN: This is Steve from Video.js. So it kind of depends. From my perspective, between the browser and the player. It depends on the specific feature we're talking about. But control specifically, we are all building those controls ourselves. And the controls differ between browsers. So anything, when it comes into controlled accessibility, that's on the shoulders of the player developer.

With captions today, we can lean more on the browsers. The latest versions of the browsers have really good caption support. But not all the browsers support all the features, and so in

some cases, we may use our own implementation of captions. So it really does kind of come down a feature-by-feature situation.

TERRILL THOMPSON: This is Terrill with Able Player. I know that kind of throughout the lifespan of our product, we looked a lot at how well browsers support accessibility features. And even just recently, I posted a blog post at terrillthompson.com, sort of giving the latest on support for various features.

And so sort of my vision all along has been the browser should support at least the stuff that's in the HTML5 spec. And so yeah, it would be great if we had that across all the browsers. And so I've always been sort of looking with the question of, do I need to keep putting time into Able Player, or can we just rely on browsers to do this? The other piece of that is every browser does it a little bit differently. And so if you want a website that has a look and feel that is consistent across browsers, then having that third-party player, that does play a key role in that.

STEVE HEFFERNAN: To answer the JavaScript question really quick, if security does block JavaScript from being loaded on a page, then I mean, your web experience is extremely crippled at that point, because so much of the web depends on JavaScript. So I see rare occasions that that's actually the case. But if it is, you can fall back to the browsers' built-in HTML5 video player. But in the cases where I do see that level of security, you're also often on an older version of Internet Explorer that doesn't support HTML5 video. And if you're not using JavaScript, it's likely you're not using Flash. So you're just probably not watching video at that point, if that's your situation.

GREG KRAUS: And so I have another question for the panelists, a little bit shorter answer here, maybe. Of the video players that you support, there's been some questions about what types of integrations are available for integrating things like Drupal, or WordPress, or other content management systems. And what abilities do you offer there, so you don't have to go to some other-- even put it in your own content?

ERIC BOYD: Sure, this is Eric from JW Player.

MATTHEW SCHWEITZ: We deliver-- oh, go ahead, Eric.

ERIC BOYD: No, you can go ahead. It's fine.

MATTHEW SCHWEITZ: I was just going to say we offer an embed player that has a pretty extensive API that you can embed in any context, regardless of the content management system or blogging platform that you're using. This embed player should have all the same accessibility controls that our normal on-site player has. And you can find out more about it on the Google Developers website. Just search for the YouTube Player API.

ERIC BOYD: Yeah, I was going to say the same thing. JW Player can be embedded in any CMS. We do have a WordPress plug-in to do seamless integrations there, and we also offer a WordPress VIP login. So that's WordPress certified. And our API, you can use that to integrate it into any homegrown CMS, Drupal, anything like that. And it does contain the same accessibility features that are available from a regular cloud-hosted player.

STEVE HEFFERNAN: Yeah, for most video players, as long as you have a way to input HTML, you should be fine dropping in any of these video players. Sometimes it's nice to have-- like WordPress has their tags interface. That's a little bit different than HTML, and you need a plug-in for that. But even in that case, I think there's still an HTML mode that you can use. So there's a few different options, and there'll be different plug-ins for different platforms for the different players.

TERRILL THOMPSON: And my answer to that-- this is Terrill-- is actually hot off the press. On our issues list has been made both a Drupal module and a WordPress plug-in, and we just, last night, got somebody to step up to start working on, and have already started working on, the Drupal module. So we're really excited about that. And again, it's open source. So anybody on the webinar who's interested in contributing to that, please do so.

GREG KRAUS: So, I've got a question for the panel. When we're talking about video player accessibility, how does this relate to the mobile experience of video, whether it's an iOS/Android device? The stuff we're talking about here today, how does that relate to the mobile devices? And are there other things we need to consider when delivering on mobile devices?

STEVE HEFFERNAN: Yeah, this is Steve from Video.js. Mobile devices are definitely a different beast altogether, because in a lot of cases, we don't have control over building the controls. At least specifically on the iPhone, and I believe even Android, going to full screen, we can't overlay our own custom controls. So we're really relying on the browser to provide the accessibility at that point. Yeah, I don't know if any of the other guys want to talk about it, but that's a real challenge. But a lot of those players actually have decent accessibility, so I actually can't speak to how big of an issue that is myself.

TERRILL THOMPSON: I would just second that, that a lot of the features that we've added in Able Player, things like the interactive transcript and the second video of sign language, those are basically just ignored within iOS, at least within the iPhone. Once you launch the video, then that opens up in the iOS player, and so all those features are lost.

MATTHEW SCHWEITZ: Yeah, I'll second what folks are saying, from YouTube. We are trying to pay a lot more attention to the mobile accessibility experience. Right now we've tried to cover a lot of the basics on iPhone and Android, in terms of using VoiceOver and the Android accessibility affordances. It's a bit more of a challenge in the mobile web situation, especially trying to deal with all the different platform nuances, in terms of video playback and interacting with native playback on the web, and various platforms taking over playback. It's something we are trying to invest a lot more in.

GREG KRAUS: So another question for the panel. So when do we know that we've gotten there with accessibility? And I mean that in two senses. I mean, there's the sense of when have we actually gotten there, and that maybe it's as accessible as it's ever going to get? But then there's the realities of shipping a product and the iterative process of that. So how do we know that we're there for a particular version? Where we draw the line to say, it has to ship now versus we need this feature?

MATTHEW SCHWEITZ: This is Matt from YouTube. This is a tough question that we think about a lot. In the end, what we're looking for is for accessibility to be a part of the conversation as early as possible in the sort of feature or fixed life cycle. So we'd like to get to a point where every user research study, every design exercise, every prototype, has some thought about accessibility in it, sort of at the earliest possible stage, in terms of development.

VLADIMIR VUSKOVIC: Yeah, and another point of that-- this is Vlad from YouTube again. So we really want every video on YouTube to be accessible. So you know, we can think of being there when we are able to perfectly-- to create perfect autogenerated captions for every video on YouTube.

MATTHEW SCHWEITZ: So a big goal, but we'd like to get there.

STEVE HEFFERNAN: Yeah, for Video.js, we kind of get our requirements from two different places. One is standards, like WCAG, and the other is just people submitting issues on the GitHub repo and telling us what they're expecting out of the player.

And so the player, you know, we're releasing new versions of the player every month, so it's kind of an iterative process. And as people add more accessibility features, they get released pretty quickly. But I guess when we're there, for me, would when we see fewer and fewer of these issues coming in requesting accessibility features, and when we can kind of check the box and all of the different items on the standards like WCAG.

TERRILL THOMPSON: This is Terrill with Able Player. I don't know that we'll ever be there, where we can say, OK, that's accessible enough. We're finished. Because it's just like any other feature set. You know, we're never going to be finished with our players and say, OK, that's it. We don't need to do any further development, and accessibility's just another feature.

But also, we've referred quite a bit to a document, editor's draft within the W3C, called Media Accessibility User Requirements. And that was an effort to just lay out all of the different user groups, particularly focusing on accessibility, and what the needs of those user groups are, and sort of what their user requirements are related to interacting with video. And so we actually discovered that sort of halfway through our process of laying out our vision and where we wanted to be. But it has been a really great resource to compare what we've done with where we want to be. And there's a lot of stuff in there that, realistically, probably will never happen. But it is something to be aware of and a good resource to consult.

ERIC BOYD: Yeah, I agree with you on that, Terrill. I don't know if we'll ever actually be there. But at least coming up with a definition of what is that "there," what is it that everybody who's attended this webinar, what does that "there" mean for you? And filling in any of those gaps, checking those check boxes, to see if it works. You know, for us at JW Player, we have some things in mind of where we want to go, and want to get better screen reader support, but that doesn't necessarily mean will be there even after that.

And there's also-- there's new media formats coming out. There's new devices that are going to be supported. New video experiences, you know? What does accessibility mean for a 3D video experience, and how do you interact with that?

So it's always a constant research, constant interaction with our users, and really just trying to make the best possible experience. And then also empowering publishers, that if the player doesn't by default provide that experience, giving them the ability to adjust the player to their needs.

TERRILL THOMPSON: Yeah, another way to define "there" has to do with the content. And I think, yeah, it's great that we're all building accessibility into our players. And the other players that you mentioned, Greg, that have accessibility features, it seems like there is a lot of work going into players that support accessibility.

And so to borrow the cliche from *Field of Dreams*, "If we build it, they will come," that really is our hope, that if you've got all these players that support captions, for instance, then we would love to see that everybody's captioning their video or finding some way to do that. And hopefully, you guys at Google and YouTube can continue moving the science forward, in terms of automatic captioning. But also, hopefully, we can find some creative ways, maybe with you guys, 3Play, to get cost down to where everybody can afford to caption everything.

I'd love to see all video everywhere captioned, and then just all the stuff that can come with that, of interactive transcripts on all video, and fully searchable video. But at this point, I know in higher education, we're really struggling to meet that, because it's expensive. And so we have to be really selective in what we can caption.

GREG KRAUS: OK, well, we're coming up to the top of the hour here. And I want to thank everybody on the panel for participating today and for the great information that you're able to share about the approaches you're taking and some of the directions that you're going in. Just so people are aware, there were a whole bunch of questions that came in about YouTube's new stuff they're announcing with captioning. So there's a lot of exciting things here. I wish we could have gotten to all the questions, but I'm going to go ahead and turn it over to Lily now for wrapping things up.

LILY BOND: Thanks, Greg. I want to reiterate, based on a lot of questions, that we have recorded this presentation, and we'll send out an email with a link to the recording with captions, as well as the slide deck, shortly. And I want to thank everyone for joining. Thank you, Greg, for moderating, and thank you to Matt, Vlad, Eric, Steve, and Terrill. You all had wonderful presentations, and everyone has really appreciated them. So I hope everyone has a great day. Thank you.

MATTHEW SCHWEITZ: Thanks, Lily and Greg, and all the other panelists.

VLADIMIR VUSKOVIC: Thanks, everybody.

STEVE HEFFERNAN: Yeah, thanks, everyone.

ERIC BOYD: Thank you, everybody.