WECO ARIA BASICS

Wednesday, December 4, 2019

TRANSCRIPTION PROVIDED BY: CAPTIONACCESS

Female Speaker: Good afternoon everyone. My name is Kelly Ryan and I am the director of operations and I will be moderating the webinar. You can email me at kelli@theweco.com. I am going to hand this over to Dane.

Dane: Thank you. I am Dane Dunham and I am an accessibility specialist and I will be instructing this webinar today. It looks like it is 12:00 so we will get started. Go to the next slide Kelly. It should be slide 3.

I am going to go over what we are going to cover. We are going to cover basic concepts of what ARIA is and how it works and the components and common ARIA attributes and some basics do's and don'ts of usage. We will go over WeCo as well.

What is WARIA stands for web accessibility initiative accessible rich internet applications. It is called ARIA because it is a suite of standards that is developed by W3C. It allows developers of websites and web apps to provide the interface elements in the applications to assist with technology for people who live with disabilities.

It allows developers to define how their elements should be defined running on the user's device. I like to think of it as a way to tell assistive technology what the intention is behind the controls. ARIA doesn't create any interaction between the user and the application or the user's technology in the application. That is something that needs to be done by the developer with JAVA.

That is what ARIA is. ARIA changes and augments the standard DOM accessibility tree. For those who don't know the accessibility tree is computed by the web browser or other application such as a web app but usually the web browser if you are talking about a laptop or a PC. It tells us what elements are on the page and how they act and so on and so forth so it can read and interact with elements like screen readers.

It lets you create with type elements that wouldn't be possible with just plain HTML. The ARIA attributes don't affect anything about the webpage. The only thing that changes is the information that is with the API which is where the accessibility comes from and where it gets its information from.

This is a common question that we see. Is ARIA supported? It is difficult to answer that because we have so many features in the ARIA specification and not all of them are supported but many are. It is an evolving specification. There are also so many different combinations of web browsers and operating systems and it can be hard to determine how everything will act in the real world.

On top of that some HTML elements have restrictions on what type of ARIA attributes you can apply.

For example a hyperlink can't have the state of selected applied to it because they don't select anything. They take you to another page or a place on a page. It can be complicated to figure out the exact support of ARIA.

This is an overview of what we will talk about. We have three main features: Roles, properties and states.

ARIA roles:

They define what elements are or what they do. A lot of these are landmark roles which tell system technologies what the structure of the page is and duplicate a lot of HTML 5 structural elements. These are things like navigation and different page structures like banner search which you would use on a page with a search field.

There are others and we won't go into all of them today but there is content information that tells the technology you are entering into the footer of a page. There is link, check box, heading, etc. and those things tell the assistive technology what type of element it is. A lot of them have HTML that covers that but we have them in cases where that can't be used.

We have tab to create things like tab widgets. That is what roles are.

Properties:

Different aspects of elements which tells assistive technology extra meaning or semantics about an element. An example would be the ARIA hyphen required which tells a formed input needs to be filled it. It doesn't create the validation or an actual requirement that will prevent the form from being submitted. It just tells of the intent.

We also have the required attribute at HTML5 that does the same thing. We have ARIA hyphen label which allows you to assign a label to an interactive where it doesn't have a visible label. If you are doing a tool bar in an interactive editor and here is a list of some common ARIA properties. We won't go into them in detail. This is ARIA atomic, controls, described by, pop-up and label, labeled by, ARIA live, ARIA owns and ARIA relevant.

We won't go into those in detail but wanted to give you an idea of those.

States:

These are special cases of properties that we use to define current conditions that apply in elements. ARIA true tells assistive technology that a form input or button is disabled. It doesn't disable it but it tells it that it is so you would use JAVA to apply that when it gets disabled. You would tell assistive technology that it has become disabled. It is the same thing with a text field.

States differ from properties where they don't [Inaudible.] where states do. Here are some examples: Aria busy, aria hidden, aria invalid - next slide.

We will talk about the most commonly used ARIA attributes.

Landmarks:

These provide information for screen reader users to help them find common page elements. They separate the page into regions so on a nav element you would have things like "less than Nav" and "role equals navigation" and that tells the screen reader it is a navigation reader. You can apply a search to tell a reader this is a region with a search form.

You can do the same thing on a development at the bottom of a page to put the info and you could associate that with a footer element in HTML. There are a few more but we won't go into all of them today.

Dynamic content updates:

These days it is a lot more common for websites to be more than static text and hyperlinks and static unchanging graphics. That was more common in the early days of the internet. They have a lot of this dynamic content that updates without the entire page reloading it can be an issue for a lot of assistive technology users. It can impact speech software who might not know the content has changed.

In that case we have in ARIA a special property that can be used to inform assistive technologies. If you apply the ARIA live to an element it will cause the screen reader to read out what has changed. How urgently that is communicated depends on the value of the ARIA live and how it is set in the code.

We have off which is the default when ARIA live isn't present. That updates them to not be announced. Polite is when the user is present and assertive is when they will announce as soon as possible and interrupt the user with the change. This is something we have seen misused. We see this happen a lot where developers will create an interactive web app from an email application to a database system to an advanced search widget.

They will put ARIA live on the entire container which makes it announce Everything when it changes which is confusing. You would put it on an element that contains an update. We have a status role which you can use to place on an update for a search field. There would be a message that says 30 results displayed and that would be the only element that would have the ARIA live. The screen reader could go down and look for them.

The ARIA describe by can be used for missing information or information that could be missed by screen users. If you are interacting with a form field you use the tab key to move from input and if you have a hint for a user because paragraphs of text that aren't focused the user will miss it so you would provide a unique ID for the paragraph of text and bullet the ARIA and reference the ID of the hint which would allow the screen reader to read it along with the label with the form field.

Adding context - sometimes we have an interactive element like a button that doesn't have any text associated with it. It might just say button or not be able to tell the user other then the type of control so the user wouldn't know what happens. If you add a label you could add BOLD or a help icon or a close button at the corner of a window that is represented with an X.

A lot of times it will say X button because that is the character that is being used. That will allow the screen reader to announce that.

We are going to go over some basics dos and don'ts of ARIA use. These are examples of using it correctly. When you develop dynamic apps keep in mind how users including those living with disabilities will interact with the controls. You would do a web mail application and you would have a tree view on the left side that shows folders. Across the top is a tool bar with compose, archive, forward, etc.

The messages are the rest of the screen state. You could use ARIA to divide the screen into regions based on those areas and the folder preview tool bar and message list if it is divided you could help them navigate between sections of the page. It is usually from region to region and since HTML doesn't do so then ARIA could be used to inform the assisted technology that it is a tree view and it can be collapsed and then you would use JAVA to show the actual interaction.

ARIA could be used to tell users navigation is being used or collapsed. MOTO windows are also helpful because it can be used for this which is referred to as a dialogue along with the windows title so the user can understand the results on the page.

This is incorrect ARIA usage. It can be beneficial but it has some drawbacks. ARIA labels can be applied to any elements on the page but when applied unnecessarily they can override visuals that are being displayed which can prevent readers from accessing all information on the screen. You could have show product instructions with an ARIA label and they thought they were adding a hint but because the label overrides anything displayed it will cause issues for users. It would just say air conditioner link instead of [Inaudible.]

It relies on developers for the interaction behind the scenes. They have to keep track of those and change those with JAVA.

In some cases we have designers or developers that use custom coded controls even when check boxes or other elements would fit the design. ARIA can be used to provide accessibility which might work for some but custom controls can leave more room for mistakes and oversights which can lead to inefficiencies or leave it impossible for users to use the website.

I will summarize the content. Even though it can be useful you should be careful of when and how to use it. Use semantic HTML5. Don't alter the meaning with ARIA roles. Interactive ARIA elements must be accessible via keyboard and not just mouse. Use appropriate roles and interactive elements must have an accessible name.

We are going to go over some basic resources here at WeCo. You can learn more with us if you would like.   Feel free to go to our events page to get information on our upcoming webinars and workshops. If you need something special we can offer customized training. You have visit the resources tab where you can find our library, workshops, webinars, conferences we might be at or online.

We also provide you with some free and easy ways to stay on top of accessibility. We have a blog and a few times each month we can notify you via email about our blog and how users with disabilities interact with IT. We have everything from accessibility experiences from our testers to latest legislative updates. Our emails provide you with tips that have some informative videos to keep your websites and software accessible.

It helps you keep these things first and implement it. Feel free to use us. You can find us on the web at theweco.com. You can contact the accessibility team at accessinfo@theweco.com or at 855-849-5050, extension 1.

That is all I have for you today. Does anyone have questions?

Female Speaker: There are no questions from the chat box.

Dane: I can . . .

Female Speaker: I do have one.

Dane: That can be tricky. It depends on what the content and purpose of the live regions are. If it is to update the users of basic changes like the example polite or assertive - I got it - that would be true. In most cases the polite would probably be the better one to use because it gives the user more control.

I would only use the assertive one in my professional opinion in a case where it is absolutely urgent. If the user is about to delete some important data and you don't want them to miss the notification but you could use a MOTO window with a role of dialogue. It is hard to come up with an example where the assertive would be most appropriate. I would agree polite would be the best one to use. I hope that answers your question.

You are welcome. We are at 12:30. I will conclude today's webinar. Thank you for attending. I will conclude.

Female Speaker: Thanks Dane.

[End of Meeting.]