Beyond WAI-ARIA - Accessible Interactivity

Transcript from the 2017 Accessibility Conference

Jason Tarka, software Developer, D2L

For more information, contact:

http://tarka.ca
a11y@tarka.ca

Transcripts are available courtesy of Ai-Media who provided live captioning at the 2017 Accessibility Conference.

Note: The following text is taken from a live transcription of the speaker's presentation and, as such, may not be wholly accurate. Please contact the speaker first before publicly attributing remarks to them based on this transcript.

SPEAKER:

OK, we're going to get started. It is 3:00 and you should be here for WAI-ARIA. Jason Tarka always knew about website department, and was disappointed about the work of accessibility. He is now trying to share as experience and teach others what he meant. Please join me in welcoming him.

(Applause)

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License - https://creativecommons.org/licenses/by-nc-nd/4.0/
Transcripts provided by Ai-Medi.tv
The 2017 Accessibility Conference, Open Learning and Educational Support, University of Guelph, Guelph Ontario Canada N1G 2W1 www.AccessConf.ca

JASON TARKA:
Today I'm going to talk to you about going beyond WAI-ARIA. I'll be going over a little bit what this talk is about, some of the general techniques that I've learned, things that I've implemented and some of the challenges I've come across.

To start, what is accessible interactivity? What do I mean by that? I'm talking about controls and web applications that have interactive controls. I'm not talking about just the platform. The controllers can change the meaning of the page. Hopefully there styled as well. Talking about mostly keyboard interactions and how useful they are.

Why am I talking to you about this? We have rubrics that teachers can use to evaluate students. We have a screenshot here of rubrics that used to be used 10 years ago, it wasn't very useful. But then we made a new one. We struggled with making it accessible, but we tried to convince ourselves that it would be better than the old one. We were wrong. We gave it a try and it was completely unusable. I had no idea what was going on. And this is despite having looked at it for the past 8 to 10 months.

So I decided to completely overhaul it. It took me a month or 2 to almost completely rewrite it. Some of the things that were more advanced or interactive had no information online. It became very hard to troubleshoot. It lead to a lot of questions and challenges. Some of these challenges took hours, some took days. When I was done, you can see the screenshot of the rubric. In case you cannot tell, the rubric is exactly the same. But a lot of how it works is changed.

Context is key. A lot of the interactions and how you're going to work with it is based on this. If you're coming across a particular control, how do you interact with that? How do you navigate around a page? If you have something custom and you're doing something fancy, it is not always going to be obvious how to work with that. If you are using a screen reader for example, you might not necessarily know how to use the most effect of the in a particular scenario.

Viewing the rubric you can tell immediately that there are groups of criteria. But if you're using a screen reader you don't have this information. For even more complicated controls that aren't necessarily obvious how to use, you can provide short instructions on what it is, and how to use it. This can be done with in offscreen text or a label. If you put an ARIA label on an area, it can read it out without interrupting the page.

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License - https://creativecommons.org/licenses/by-nc-nd/4.0/
Transcripts provided by Ai-Medi.tv

We have HTML here showing some information. This immediately provides information on how many groups there are, the fact that they are in groups.

Faking visual focus is something that can be used to provide extra semantic meaning or semantic ordering, extra information to elements. It can be interacted did with, but keyboard and screen reading element consider it off screen. Someone who is tabbing through a page may not know any different.

This will likely lead to duplicating information. So you want to make sure you hide the on-screen element. I will show a bit of that in a moment. Here is a screenshot taken from rubric, and you can see we have the criteria here on one side. Next to it you have four levels you can choose from. If you're on the screen reader and you're looking through the page, you would expect to encounter anything with the feed back right after the levels. But for technical reasons we needed to have the feedback button is placed within the header, to have it displayed correctly. This didn't make sense since to get to the feedback you would skip past the levels. This is not helpful. So we changed it.

If JavaScript and CSS for this, if you have your title and your score on criteria one, and the actual display there, and for technical reasons they may have to be in the same cell, then the off-screen elements in another element. The JavaScript for this is very simple. The user may not know that the offscreen element exists, but it works correctly.

If you're going to be moving focus you need to make sure that any time you move that it is for the purpose of maintaining the user's context. Even the smallest mistake in places can have huge effects, huge issues. You can lose focus on the page or you can cause the user to be very confused. You need to know the impact of moving the user’s focus. If you're jumping around the page a lot, the user can get easily lost. It is also important to make sure that everyone understands why you are moving. Unless you need to move the user’s focus, don't. It is something you should only use if you absolutely need to.

Grouping elements can help add extra context, and this is particularly useful for screen readers. You can add extra information that will be read out or shown in a form. You could have several customer information fields, where a customer is tied to it. Particular screen readers will let you jump between groups on a page, helping you find exactly what you're looking for quickly.

Having temporary text is a way of working around issues on ARIA live, which I'll explain shortly.

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License - https://creativecommons.org/licenses/by-nc-nd/4.0/
Transcripts provided by Ai-Medi.tv

You can insert extra elements that state what you need. It gets read immediately and then the rest of the text gets read.

A potential issue with this however, is that if you are focused on the temporary element and you remove it when the user moves away from it, it can cause the focus to be lost. In at least one or two screen readers I tried, it caused serious issues. The way to get around this is to add a timeout, just a couple of milliseconds.

Again, a little bit of JavaScript and HTML on how can do this. You show the button, you show the text and then add a word handler. You can then set a time out and remove the text. When the user tries to scroll back, the text will not be there because it is no longer relevant to them.

When I first started doing an overhaul of the rubrics, I ditched a huge amount of the CSS we had, and added this other monstrous mess, which you can see on screen. It helped me focus on the functionality without worrying about how it looks. It is easy to get caught up on the little details of how it works. But doing it this way forces you to see things the way you user sees it. If too much is blacked out, or not enough is blacked out, you might have some issues.

If you click on a field or activate a field in any way it brings you to edit mode. People using a mouse can figure it out by the area changing colour and the cursor changing. They can play with that and there are no side effects to it unless they change something. It is a great way to learn ways to interact with it. It is important to design it in a way where users can take in information and not get overwhelmed by the fact that everything is editable.

It is a good idea to add an offscreen button. Having a button there will let things show up in a screen reader. If it is in a form it will try to submit the form, unless you set type=”button”. The on-screen element you want to hide on the screen reader, because it is duplicated information and you don't need it. When you focus on the button you made the on-screen element your focus. You want to duplicate the actual title and have the styles the same so it is very seamless. Having the label there will allow a larger click target. You can also have instructions there on how to change things and how to get out of it. The instructions will be read out and the contents of the field will be read out after.

When the user is finished working with that or finished editing, you want to update the on-screen and button text. Depending on what triggered the actual closing of the editor, you may want to focus on it. If you're listening out for certain keys, you want to focus back on the

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License - https://creativecommons.org/licenses/by-nc-nd/4.0/
Transcripts provided by Ai-Medi.tv

controls. If a user has moved their focus then you don't want to focus back on it, otherwise you will interfere on the user's context. Adding something to alert a user that they have finished editing helps them keep context.

Some HTML here of a click to edit control. Unfortunately the JavaScript is way too long to fit on the couple of slides. But you have an offscreen element which in this case states address. Then you have your on-screen display which is hidden from screen readers and indicates address and values. And then initially hidden, you have a label within an input, and you can put in the field. When you activate it, you focus on there. It has instructions to hit tab to finish and escape to cancel.

In a rubric you will often have several levels for criteria. It may have a couple of paragraphs of description for each one. However, instead that information is not going to be meaningful most of the time. If the user is very familiar with that, they will know the specifics. They don't need to hear it every time. If you have to hear the description every time, it can get very repetitive, especially if your screen reader decides to read all of the text before letting you know it is a radio button. If you have to hear that every single time you get to that button, you will get really annoyed and shut off your screen reader.

What if I told you there was an easier way? You can configure it so read that only the title, and don't need JavaScript. You need to duplicate the title on the overall label. You want to hide the duplicate name again. If the user wants to hear the details it's easily available, however if they don't need it, they don't need to hear it. I have information he showing you how to do this.

You have your input, you have your standard title, and a full description. But inside the input you have ARIA labels. So when jumping through, you will only hear the titles. That is a lot easier to digest and flick through. The description is still there and easily accessible.

Modal dialogue is something I have not worked with a lot but people have asked me to talk about. I'll go over it quickly. Whatever the container for the dialogue is, you need to set it as a dialogue. Modal dialogue is something that pumps up and takes over exclusive control of the page. An example of this is a JavaScript alert, where the only thing you can do is click on it. The rest of the page is not accessible.

You also want to make sure that user's cannot leave that dialogue. So as they are navigating through, if they hit the end it will go back to the beginning. Doing this prevents them from

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License - https://creativecommons.org/licenses/by-nc-nd/4.0/
Transcripts provided by Ai-Medi.tv

messing with a page when they are supposed to be in a modal dialogue. If they clicked the button to open the dialogue, then focus on the same button when closing it, if you can. Otherwise, designs and very similar. I've put together a demo, and I will have a link to it at the end of the presentation.

We're going to try a live demo here.

(computer reads information)

And this is a heading here. John Smith is a heading here that is designated to a field set. However, having it with a header as was a grouping allows you to quickly jump to a section if people are navigating through headers.

(computer reads info)

And this is an example of the data field... So this is an example of the click to edit field for a first name.

(computer reads info)

There is the offscreen button, and it looks like it is focused on the screen. I activate it...

(computer reads info)

And you'll see that it reads out the information. I need to change the demo but I used an ARIA live here. It would work better with some temporary text.

(computer reads info)

And you can hear that there is a bit of text that is not visible on screen. This gives you some information in advance. And if you know what a given customer type is then you are good. If you don't, you can access the description.

(computer reads info)

And then we're getting into the actual radio group here.

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License - https://creativecommons.org/licenses/by-nc-nd/4.0/
Transcripts provided by Ai-Medi.tv

(computer reads info)

The full description is available if you need it. However, if actually activate it...

(computer reads info)

So as you can see, as you're jumping through it, all we hear is the title. If you need the description, you can switch the mode.

(computer reads info)

And then everything you need is right there.

The question was to have to do a lot of work with screen readers to make sure they work correctly? Yes. The main ones I worked with were NVDA, Jaws, and Voice-Over. I also gave it a try with Orca, and despite it being buggy in general, it did work with this.

Some of the challenges that I had with controlling user focus while going through the process. Screen readers will read the element on the screen immediately before focus change. If it is changed with JavaScript it world jump immediately. It can be confusing. There is one situation I came across, and you may come across as well, was focused on an outer element and based on the interaction with that I needed to jump onto a child element. One of the screen readers was having serious issues with that. It took me ages to figure out what was going on. I needed to have another element with nothing in it to make for a smoother jump.

The order of focusing on events can matter, and they are different depending on how it is generated. It can be generated either user clicking on another field or hitting the tab key. This means for a split second that nothing has focused. If it is generated by JavaScript than that focused event happens before the blur event. This can have a big impact. The location on the page can easily be lost if whatever you're focusing on is hidden, or removed from the page. And this can have serious side effects. Either nothing on the page appears focused, which for a screen-reader user is going to be read, or the focus will be redirect did to the top of the page and you will have to tab through again. In one instance, it made a page completely unusable. If you're removing elements, make sure you shift the focus to somewhere else before removing it

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License - https://creativecommons.org/licenses/by-nc-nd/4.0/
Transcripts provided by Ai-Medi.tv

when possible.

Removing or hiding an element when it blurs can cause similar issues. You have to add a slight timeout to let it focus on a new element before the old one is blurred. You have to test with a number of screen readers to make sure that it works. When you are dealing with interactive controls, you don't know how everything will work.

This is something that can happen when you're trying to control the user's focus. If there is a case you want at exit something based on the user losing focus, you want to hit the tab key to exit instead. This is a good way to maintain focus. Make sure you use that key down event and not key up. Crucially, if the user clicks somewhere else or they move the focus away from the elements, don't move it. It could be you are one place, and the user clicks something entirely different. If the focus is lost, then you're suddenly jumping around and the user will get lost and likely extremely frustrated.

One issue I ran into is if you are only reading changes, I had it change from “proficient” to “partially proficient”. Not a lot of the text had changed. But the effect was weird. You need to make sure you hide this sort of thing using ARIA hidden if it is not something you want the user to focus on. If you start something and the focus is shifted, it stops reading the ARIA live message. This is where temporary text comes in handy. You can have an element, have some text in it, and some JavaScript. This will read out the changes but the entire contents are blank. When you put the text in, it is entirely new text, everything is changed so everything will be read out.

A couple of miscellaneous things, you need to make sure you set the role=”textbox”, or the user will not know that they can enter edit mode. We had an issue where one screen reader would enter edit mode, but another would not. If you are using iframes... You could tab onto it. Make sure that you don't mess with this. If you cannot set a tab to it, then you cannot tab into anything inside it, and it might as well not exist to the user.

In conclusion, you do not need to sacrifice design for function. As long as you take care and how you are doing it and test the changes as you go, it can be fairly easy to make it accessible. You want to show the correct content to the correct audience. If you're showing something to a screen reader you may provide extra information, because they need it. If it is meaningful content, you need to make sure you can include that content for everyone. Pay close attention to how your controlling the user's focus. It is vital to get this right.

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License - https://creativecommons.org/licenses/by-nc-nd/4.0/
Transcripts provided by Ai-Medi.tv

I have here a few resources if you would like to take a look at them. Are there any questions?

QUESTION FROM FLOOR:
Will you be providing your slides to us?

JASON TARKA:
I will have information available on my blog.

SPEAKER:
You can also send Jason an email. Information is available on a website.

QUESTION FROM FLOOR:
Was that a framework that you made, rubric?

JASON TARKA:
We used AngularJS, but everything else was something we created. It was fairly straightforward. We were able to add a plug in, ngAria. There were a lot of smaller problems that we had to figure out ourselves.

SPEAKER:
Then please join us in thanking Jason Tarka.

(applause)

Thanks, everybody.

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License - https://creativecommons.org/licenses/by-nc-nd/4.0/
Transcripts provided by Ai-Medi.tv
The 2017 Accessibility Conference, Open Learning and Educational Support, University of Guelph, Guelph Ontario Canada N1G 2W1 www.AccessConf.ca