State of ARIA 2018

Joe Humbert

Senior Accessibility Specialist
Interactive Accessibility, an Affiliate of The Paciello Group

Overview

1. What is ARIA (and what it is not)
2. Current level of support
3. What’s next for ARIA
4. Testing ARIA support

What is ARIA…

…and what it is not

Note: this presentation is not endorsed by the W3C®

> **WAI-ARIA, the Accessible Rich Internet Applications Suite, defines a way to make Web content and Web applications more accessible to people with disabilities. It especially helps with dynamic content and advanced user interface controls developed with Ajax, HTML, JavaScript, and related technologies.”**

– **WAI-ARIA Overview**

Accessible Rich Internet Applications (ARIA)

- WAI-ARIA is part of the Web Accessibility Initiative (WAI) standards & guidelines
- Current version: WAI-ARIA 1.1
- W3C recommendation on December 14, 2017

Breaking Down ARIA

- Provides an ontology of roles, states, and properties that define accessible user interface elements and can be used to improve the accessibility and interoperability of web content and applications
- Specifies how the different roles, states and properties can be used together and how they relate to one another
- Supported by other documentation:
  - WAI-ARIA Authoring Practices 1.1
  - Core Accessibility API Mappings 1.1
  - Accessible Name and Description: Computation and API Mappings 1.1
  - ARIA in HTML

What ARIA is Not

- A silver bullet
- ARIA solves the specific need of providing semantic information for HTML elements where native semantics are insufficient for accessibility
- ARIA is not intended as a substitute for existing best practices for accessibility
- ARIA by itself is insufficient to satisfy all the requirements of accessibility for the many rich controls found in modern web applications
- ARIA “lies” to the user; it changes how a screen reader recognizes the control, but not its behavior
- ARIA isn’t without its own set of rules

What's the First Rule of ARIA?

*If you can use a native HTML element [HTML51] or attribute with the semantics and behavior you require **already built in**, instead of re-purposing an element and adding an ARIA role, state or property to make it accessible**, then do so**.*

Using ARIA

https://www.w3.org/TR/using-aria/#rule1

Don't use ARIA.
(When you can avoid it.)

The Five Rules of ARIA

1. Don’t use ARIA
2. Do not change native semantics, unless you really have to
3. All interactive ARIA controls must be usable with the keyboard
4. Do not use role="presentation" or aria-hidden="true" on a *visible* **focusable** element
5. All interactive elements must have an accessible name

Current Level of Support

Disclaimer

- The following resources that were reviewed are not owned by me in anyway shape or form and they stand on their own
- The resources reviewed are NOT inclusive or exhaustive
- I want to thank those who created the resources for their efforts
- I spot checked some of these results with current versions of AT

Caveats

- It is crucial to understand the relationship between the ARIA markup itself, the browser in which the markup is rendered, and the assistive technology used:
    1. ARIA markup was designed to insert information useful to assistive technologies into existing HTML code. But adding ARIA support to a Web page does not change the presentation or behavior of that Web content for sighted users. ARIA produces no visible effects on the page by itself.
    2. ARIA should be implemented carefully and thoroughly along with other necessary scripting. For best results, Web content authors should read the WAI-ARIA 1.1 Specification, WAI-ARIA 1.1 Authoring Practices, and other documentation carefully before adding ARIA markup to their content
    3. Different ATs' support for ARIA depends heavily on the browser being used. Most browsers support some type of accessibility API (application programming interface), and assistive technologies use the API to get information about what is presented on the screen.
    4. The quality of an AT's support for ARIA markup is inextricably tied to points 1, 2 & 3.
- Modified from JAWS ARIA Support Notes - https://freedomscientific.github.io/VFO-standards-support/aria.html#jas

Resources Reviewed

- Vispero JAWS ARIA Role Support - https://freedomscientific.github.io/VFO-standards-support/aria.html
- The Paciello Group Test ARIA 1.1 Forward - test cases - http://stevefaulkner.github.io/HTML5accessibility/tests/ARIA-tests/
- PowerMapper WAI-ARIA Screen reader compatibility - https://www.powermapper.com/tests/screen-readers/aria/
- ATHENA technologies ARIA Test Pages - http://accessibility.athena-ict.com/aria/aria-tests-index.shtml

Where are we?

- That is a very good question. Answer: Unclear
- Many resources are old and possibly not maintained
- Testing done is incomplete or targets specific AT only
- More testing needs to be done with current AT, web browsers & Operating Systems

JAWS support

- Support (Firefox ?? On windows 10)
  - Roles = 69.57% - 48 out of 69
  - States = 90% - 9 out of 10
  - Properties = 63.16% - 24 out of 38
  - Total approximate support = 69.23%
- Reference: Vispero JAWS ARIA Role Support (June 5, 2018)

Cross AT support

Combo	Versions	Reliability
JAWS IE	JAWS 17.0.2619 with IE11	79%
JAWS Firefox	JAWS 17.0.2619 with FF48	71%
NVDA IE	NVDA 2017.3 with IE11	64%
NVDA Firefox	NVDA 2017.3 with FF60	76%
VoiceOver Mac	VoiceOver OSX 10.12 with Safari 10.1.2	69%
VoiceOver iOS	VoiceOver iOS 10.3 with Safari iOS 10.3	55%
WindowEyes IE	WindowEyes 9.2 with IE11	74%
Dolphin IE	Dolphin SR 15.05 with IE11	52%
SaToGo IE	SaToGo 3.4.96.0 with IE11	25%
Average	Including older versions	65%

Reference: PowerMapper WAI-ARIA Screen reader compatibility (May 20, 2018)

Combined Support

- Incomplete data
- Could only compare JAWS (unknown version) and Firefox
  - Combine approximate support = 70.12%

What’s next for ARIA

(WAI-ARIA 1.2 and more)

WAI-ARIA 1.2

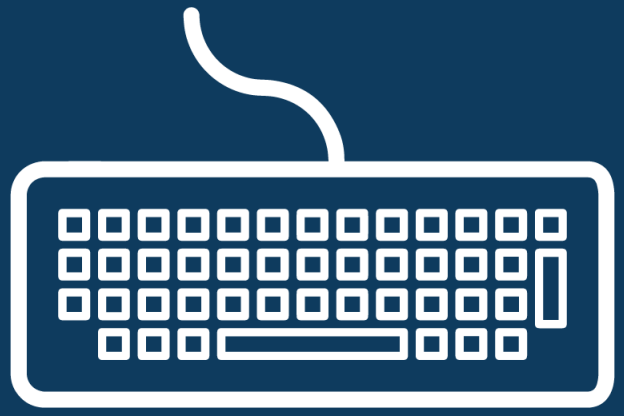
- W3C Editor's Draft 08 November 2018
- Timeline unknown for W3C recommendation status - https://www.w3.org/WAI/ARIA/project
- Minor update to WAI-ARIA recommendation
  - Add new roles to achieve parity with HTML elements
  - Add ARIA property string reflection on Element

Changes from ARIA 1.1 to 1.2

- Currently not many currently- https://w3c.github.io/aria/#substantive-changes-since-the-wai-aria-1-1-recommendation
  - **31-May-2018:** Add blockquote, caption, and paragraph roles.
  - **01-Apr-2018:** Added ARIA IDL Section (JavaScript interfaces).
  - **06-Dec-2017:** Make aria-expanded a supported state of menuitem. This change also makes it a supported property of menuitemcheckbox and menuitemradio via inheritance.
  - **06-Dec-2017:** When aria-errormessage is not pertinent, authors MUST either ensure the content is not rendered or remove the aria-errormessage attribute or its value. User agents MUST NOT expose aria-errormessage for an object with an aria-invalid value of false.

Future ARIA Work

- Achieve Parity with Native Host Language Semantics
- Provide Support for the Accessibility Object Model
- Develop and Maintain Additional ARIA Features for Authors and End Users (Authoring Practices 1.2+)
- Ensure Consistency in Platform Accessibility API Mappings
- Changes needed significant effort:
  - Develop an alternative solution to replace the now-deprecated ARIA drag-and-drop features (Target: 1.4)
  - Add support for annotations (Target: 1.4)
  - Add control patterns that enable programmatic declaration of interaction methods supported by a widget. (Target: 2.0)
  - Create means to set ARIA attribute values through platform accessibility APIs (Target: 2.0)
- https://www.w3.org/WAI/ARIA/roadmap



Testing ARIA Support

Future work

What Do We Do Now?

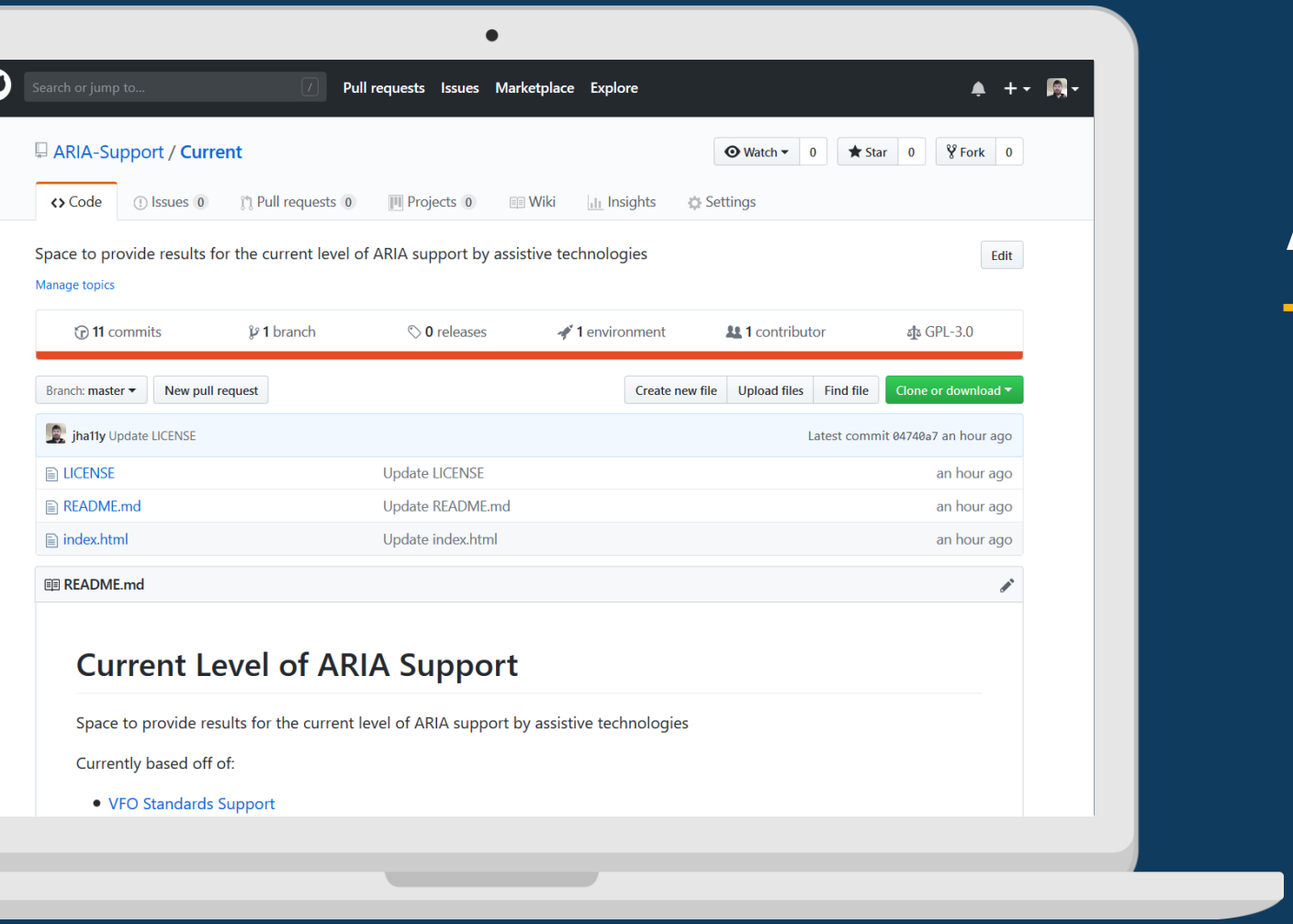
- Don’t stop testing
- Verify previous results?
- Preform new testing?
  - Solicit W3C WAI group
  - Continue independently
  - Community engagement (My choice)

Baseline requirements for testing

- Ensure that the applicable standards are followed for test cases
- Cross-browser/device/assistive technology testing
- Include complete and detailed testing records (especially date tested)
- Provide easy to read summaries
- Conduct regular testing (resources permitting)

Pseudo Test Plan

- Do not reinvent the wheel
- Community driven
  - Larger involvement
  - Enhanced verification (Tests and results)
  - More combinations
- Potentially get higher education research involved
- Suggestion: GitHub project & list serve (pseudo working group)



ARIA Support Project

- https://aria-support.github.io/Current/
- GitHub repo - https://github.com/ARIA-Support/Current
- !! Work in progress !!
- Open to comments and suggestions
- Need help

Questions


Interactive
Accessibility™
The Accessibility Experts™


THEPACIELLOGROUP