ARIA Guidelines

Introduction

HTML (HyperText Markup Language) was initially designed to create static text pages. JavaScript provides dynamic **components** which don't exist in native HTML. The **A**ccessible **R**ich **I**nternet **A**pplications (ARIA) standard provides developers with accessibility features not available in HTML alone.

ARIA is a set of attributes (properties of HTML tags) that makes web content and applications more accessible. When it is not practical to use semantic HTML, ARIA can communicate semantic information and interactions to **assistive technologies**. ARIA communicates semantic information about **widgets** (User Interface (UI) elements), structures (relationships), and behaviours to assistive technology. A proper accessible experience depends on the appropriate use of this tool: misuse of ARIA can lead to poor or unusable experiences.

ARIA attributes can make non-semantic HTML like s more accessible when native code or underlying technology cannot. For example, suppose you are using an external library or framework that uses a as a button. It may be easier to add an ARIA tag such as 'role="button" to the component, rather than rewriting existing styles and functions. Consider this approach carefully, since you will also need to confirm that there is proper JavaScript in place to make the element accessible (e.g., ensure that it is focusable and define event handlers for click and key-down events).

If you intend to enhance the accessibility of HTML with ARIA tags, please consult the authoring practices and the detailed examples below. To summarize the first rule of ARIA: If you *can* use HTML or semantic attributes**,** then do so. Another common expression is that good ARIA is better than no ARIA, but no ARIA is better than bad ARIA.

Examples of when to use ARIA

Landmarks

- Landmarks define sections of a web page, which helps **screen reader** users navigate the page. Landmarks can be defined either as HTML elements or with ARIA roles: A <header> landmark can be replaced, if necessary, on any element (like a) by adding role="banner".
- An <aside> landmark can be replaced with role="complementary".
- A <footer> landmark can be replaced with role="contentinfo".

ARIA Roles and JavaScript

Some JavaScript frameworks generate web pages with non-semantic code, for example, a element instead of a <button>. Adding the ARIA role="button" will improve provide semantic meaning to the .

When JavaScript is used to create special components that don't exist natively in HTML, they may need to be given an ARIA role to define what they are. For example:

- An accordion will be expanded and collapsed with a button which has this attribute: aria-expanded="true"/"false".
- A set of tabs within a web page will need role="tablist", role="tab", and role="tabpanel".

More examples can be found on the ARIA Authoring Practices site.

Labels

There are multiple ways to label a button – besides using onscreen text – for example:

- An aria-label attribute with text specifying the label, i.e., aria-label="continue".
- An aria-labelledby attribute can point to existing onscreen text as a label.
- An aria-describedby attribute can point from an input field to an inline error to connect them in a semantic (meaningful) way. A screen reader will automatically announce the error when the user tabs to the input field.

ARIA live regions

An ARIA live region is a section of a web page that screen readers will automatically announce if its contents change dynamically after the initial page load. The live region provides important status messages. Code examples include:

- role="alert" or aria-live="assertive" tells the screen reader to interrupt a current message to announce a dynamic content change.
- aria-live="polite" tells the screen reader to announce the updated live region after announcing the current content.

aria-hidden

The aria-hidden="true" attribute tells screen readers to ignore an element and not announce it. Aria-hidden can be useful for content displayed on-screen but confusing for a screen reader user.

For example:

- An icon placed beside redundant text (like a garbage can icon with the word "Delete" beside it) since a user doesn't need both to be announced.
- Background content that is greyed out when a popup takes over the page. When a popup takes over the page, a screen reader user won't be reading the background page.

Resources

- W3C WAI-ARIA Overview
- WAI-ARIA Authoring Practices 1.2