What is **Accessibility in web development**?

[Beginner's Guide to ARIA for Accessibility](https://www.lullabot.com/articles/what-heck-aria-beginners-guide-aria-accessibility)

When someone describes a site as "accessible", they mean that any user can use all its features and content, regardless of how the user accesses the web — even and especially users with physical or mental impairments.

Sites should be accessible to keyboard, mouse, and touch screen users, and any other way users access the web, including screen readers and voice assistants.

Applications should be understandable and usable by people regardless of auditory, visual, physical, or cognitive abilities.

Web Content Accessibility Guidelines **WCAG**:

The WCAG guidelines are a set of recommendations for making web content more accessible developed by W3C's WAI, primarily for people with disabilities — but also for all user agents.

**The 4 principles**

WCAG is broadly broken down into four principles — major things that web content must be to be considered accessible:

**Perceivable**: Users must be able to perceive it in some way, using one or more of their senses.

**Operable**: Users must be able to control UI elements (e.g. buttons must be clickable in some way — mouse, keyboard, voice command, etc.).

**Understandable**: The content must be understandable to its users.

**Robust**: The content must be developed using well-adopted web standards that will work across different browsers, now and in the future.

**What is ARIA?**

ARIA is shorthand for **Accessible Rich Internet Applications**. ARIA is a set of attributes you can add to HTML elements that define ways to make web content and applications accessible to users with disabilities who use assistive technologies (AT). When accessibility issues cannot be managed with native HTML, ARIA can help bridge those gaps.

There are **3 main components used in ARIA: Roles, States, and Properties**.

**Roles**

Roles are used to define a type of user interface (UI) element (i.e. role=“ROLE\_NAME”). Once a role is set for an element, it does not change.

1. Abstract roles

Description: While acting as the foundation for all other roles, abstract roles are utilized by browsers and should not be used in code. Instead, they are used to give roles their meaning in context and help with the addition of new roles.

How does it get used by the user? Abstract roles are in the background and shouldn’t be adjusted.

1. Document structure roles

Description: Document structure roles aren’t normally interactive, but instead provide descriptions for sections within a page.

Commonly used examples: Examples include img, document, heading, list, listitem, and toolbar (i.e. role=“toolbar”).

How does it get used by the user? Users of AT use document structure roles to identify content while navigating through a page, which helps give them the context of the content they’re taking in. While HTML 5 may provide much of this context on its own, screen reader support for HTML 5 is sometimes missing. For this reason, the most robust choice is to use both.

1. Landmark roles

Description: Created for easier navigation, landmark roles identify each section of content within a page.

Commonly used examples: Examples include banner, contentinfo, form, main, navigation, search (i.e. role=“search”).

How does it get used by the user? Users of AT use landmark roles to navigate through a page. For instance, a screen reader will announce the start and end of each landmark role set on a page, and its web rotor (i) will display a list of these roles/regions.

1. Widget roles

Description: Used when HTML doesn’t define elements, widget roles add semantic meaning to elements and user interfaces (UIs), small and large. Standalone UI widgets are part of larger, composite widgets, while composite UI widgets act as containers that manage other contained widgets.

Commonly used examples: Examples of standalone UI widgets include alert, button, checkbox, link, menuitem, tab, and tabpanel (i.e. role=“tabpanel”). Commonly used examples of composite UI widgets include combobox, grid, listbox, menu, radiogroup, and tablist (i.e. role=“tablist”).

How does it get used by the user? Users of AT use widget roles to interact with the page to get things done. This can include completing forms, opening and closing tabs and panels, and navigating a main or sidebar menu.

**States and Properties**

States and properties can be applied to elements and are used together to support ARIA roles existing on a page. States can change on their own or with user interaction (i.e. aria-checked and aria-disabled), usually with JavaScript. Properties rarely change once they’re set (i.e. aria-labelledby and aria-describedby).

Commonly used examples include aria-describedby, aria-haspopup, aria-hidden, aria-label, and aria-labelledby (i.e. aria-labelledby=“uid-123”).

**4 categories of states and properties:**

1. **Drag-and-Drop Attributes**

Description: Conveys information to AT about drag-and-drop elements, including draggable elements and their drop targets.

Commonly used examples: Examples include aria-dropeffect and aria-grabbed (i.e. aria-grabbed=“true”).

How does it get used by the user? Users of AT use drag-and-drop attributes to interact with drag-and-drop components using various methods regardless of their disability.

1. **Live Region Attributes**

Description: Indicates changes in content for a user’s AT, such as setting whether a message will be read aloud with the flow of content (i.e. aria-live=“polite”) or instead will interrupt the flow of content and be read aloud immediately (aria-live=“assertive”). These elements don’t need to have focus, and can include information on how the user can proceed.

Commonly used examples: Examples include aria-atomic, aria-busy, and aria-live (i.e. aria-live=“polite”).

How does it get used by the user? AT use live region attributes to inform users what’s happening on a page with regular messages as things change on the page. Live region attributes are also used to provide notifications to users after forms have been submitted.

1. **Relationship Attributes**

Description: These attributes add relationships between elements that can’t be determined otherwise.

Commonly used examples: Examples include aria-describedby and aria-labelledby (i.e. aria-labelledby=“uid-123”).

How does it get used by the user? Users of AT use relationship attributes to understand when information is related when navigating the page and gathering additional data from various elements like forms, menus, and tabs/panels.

1. **Widget Attributes**

Description: Used for commonUI elements that receive input from users while processing those actions and information.

Commonly used examples: Examples include aria-checked, aria-disabled, aria-label, and aria-required (i.e. aria-required=“true”).

How does it get used by the user? Users of AT use widget attributes to more easily utilize interactive UI elements, like forms and overlays/modals/popups. This helps to avoid confusion during the input process.