BE FIT MIT BFIT

TIPPS FÜR DIGITALE BARRIEREFREIHEIT

WAI-ARIA

Überwachungsstelle des Bundes für
Barrierefreiheit von Informationstechnik

Dieser Text enthält viele technische Begriffe und richtet sich hauptsächlich an Webentwickelnde und Menschen, die mit HTML vertraut sind. Wir beziehen uns im Text auf die gerade aktuelle Version von WAI-ARIA 1.2.

Dynamische und interaktive Webinhalte gehören längst zum Arbeitsalltag vieler Menschen. Doch gerade dort, wo JavaScript die Oberfläche ständig verändert, kommen Screenreader und andere Hilfstechnologien an ihre Grenzen. Hier setzt WAI an, die Web Accessibility Initiative. Sie ist ein technischer Standard des World Wide Web Consortium (W3C). Die WAI-ARIA ist eine Spezifikation der WAI und legt fest, wie Entwickelnde und Frameworks semantische Informationen ergänzen können, damit auch komplexe Webanwendungen zugänglich bleiben.

WAI-ARIA ergänzt HTML um zusätzliche semantische Informationen, insbesondere dort, wo komplexe interaktive Komponenten umgesetzt werden, für die es keine ausreichenden nativen HTML-Elemente gibt. Dazu gehören beispielsweise Menüs, Registerkarten oder dynamische Inhaltsbereiche. ARIA beschreibt Rollen, Zustände und Eigenschaften, die von assistiven Technologien interpretiert werden können.

Grundsätzlich sollte jedoch immer zuerst geprüft werden, ob ein geeignetes HTML-Element verwendet werden kann. Native Elemente wie Buttons, Links oder Formularelemente bringen die notwendige Semantik und Interaktionslogik bereits mit und werden von assistiven Technologien zuverlässig unterstützt.

WAI-ARIA kann zwar auch eingesetzt werden, um solche Funktionen auf generischen Elementen nachzubilden, etwa indem ein div als Button ausgezeichnet wird. Dies sollte jedoch nur in Ausnahmefällen erfolgen, da dabei zusätzliche Anforderungen wie Tastaturbedienbarkeit, Fokussteuerung und Zustandsverwaltung selbst umgesetzt werden müssen und Fehlerquellen entstehen können.

BE FIT MIT BFIT
TIPPS FÜR DIGITALE BARRIEREFREIHEIT

Ziel und Bedeutung

Ziel von WAI-ARIA ist es, die semantische Bedeutung von interaktiven Elementen für assistive Technologien zugänglich zu machen, insbesondere bei Komponenten wie Registerkarten, Dialogen (z. B. modalen Dialogen), Schiebereglern oder Menüs. Während HTML bereits viele semantische Elemente bereitstellt geht WAI-ARIA weiter:

» Sie ermöglicht die nachträgliche Ergänzung von Semantik, wenn HTML-Strukturen allein nicht ausreichen.

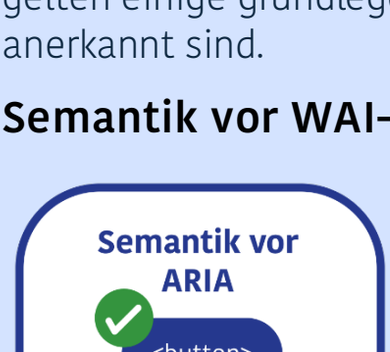
» Sie hilft, Zustände und Veränderungen (etwa „aktiv“, „ausgeblendet“, „ausgewählt“) maschinenlesbar zu machen.

» Sie unterstützt die Navigation in komplexen Strukturen, indem Beziehungen zwischen Elementen beschrieben werden können.

Richtiger Einsatz von WAI-ARIA

WAI-ARIA ist ein umfangreiches Werkzeug, aber falsch angewendet führt es schnell zu Problemen. Eingeschränkte Nutzende werden dann eher behindert statt unterstützt. Daher gelten einige grundlegende Regeln, die als Best Practices anerkannt sind.

Semantik vor WAI-ARIA



Semantisch passende HTML-Elemente sollten stets die erste Wahl sein, da sie Bedeutung, Interaktion und Unterstützung durch assistive Technologien bereits mitbringen. WAI-ARIA ergänzt diese Grundlage, wenn HTML allein nicht ausreicht, um komplexe Komponenten oder Zustände verständlich zu machen. Es ist zwar möglich, Funktionen auf generischen Elementen nachzubilden,

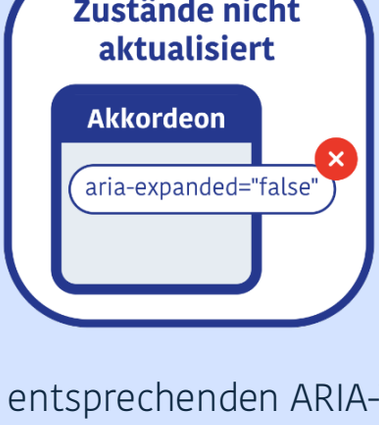
etwa durch role=“button” auf einem div, dies erfordert jedoch zusätzlichen Aufwand und ist fehleranfällig. Daher sollte immer zuerst geprüft werden, ob eine Lösung mit nativen HTML-Elementen umgesetzt werden kann, bevor WAI-ARIA eingesetzt wird.

Keine doppelte Semantik

Wenn ein HTML-Element bereits eine semantische Bedeutung hat, sollte diese beibehalten werden. WAI-ARIA ermöglicht es zwar, Rollen zu ergänzen, zu verändern oder auch zu entfernen, dies erfordert allerdings eine sorgfältige Umsetzung. Insbesondere sollten Rolle, visuelle Darstellung und tatsächliches Verhalten einer Komponente übereinstimmen. Widersprüche, etwa wenn ein Auswahlfeld als Schalter ausgezeichnet wird, können dazu führen, dass assistive Technologien falsche Erwartungen vermitteln und die Bedienung erschwert wird.

Beispiel: Ein <button role="link"> führt zu widersprüchlichen Ansagen eines Screenreaders, da die durch WAI-ARIA vergebene Rolle von assistiven Technologien übernommen wird. Während visuelle Darstellung und tatsächliches Verhalten weiterhin durch das HTML-Element bestimmt werden. Stimmen diese Aspekte nicht überein, entstehen widersprüchliche Erwartungen, die die Bedienung erschweren können.

WAI-ARIA-Attribute müssen dynamisch gepflegt werden



Zustände wie aria-expanded, aria-checked oder aria-hidden müssen den aktuellen Zustand widerspiegeln. Diese Aktualisierung erfolgt dann, wenn beim Bedienelement ein Statuswechsel ausgelöst wird. Entwickelnde müssen daher sicherstellen, dass Interaktionen wie bspw. das Öffnen eines Akkordeons auch programmgesteuert den entsprechenden ARIA-Wert anpassen. Bleibt beispielsweise ein Akkordeon visuell geöffnet, aria-expanded jedoch auf false, erhält der Screenreader veraltete Informationen und meldet weiterhin „eingeklappt“, obwohl der Inhalt sichtbar ist.

Beziehung herstellen, nicht nur beschreiben

Aria-labelledby und aria-describedby sind oft aria-label vorzuziehen, weil sie Beziehungen zwischen sichtbarem und unsichtbarem Text herstellen. So bleibt der Kontext erhalten und redundante Beschriftung wird vermieden. Aria-label eignet sich vor allem dort, wo es keinen sinnvollen sichtbaren Text gibt.

Verstecken mit Bedacht

Ein aria-hidden Attribute mit dem Wert true entfernt ein Element aus der Wahrnehmung von Screenreadern. Wird es unüberlegt gesetzt, verschwinden wichtige Inhalte, zum Beispiel ein sichtbarer Warnhinweis. Sichtbar bedeutet nicht automatisch zugänglich.

Prüfen in realen Umgebungen

Assistive Technologien interpretieren WAI-ARIA unterschiedlich. Daher ist das Testen mit Screenreadern wie beispielsweise: NVDA, JAWS, VoiceOver oder TalkBack unverzichtbar. Automatische Tools können helfen, ersetzen aber keine echte Nutzungsprüfung.

Überblick über zentrale WAI-ARIA-Attribute

Die folgende Übersicht erläutert einige wichtige Rollen und Attribute in WAI-ARIA 1.2 und erklärt ihren Zweck in einfachen Worten.

Kategorie	Attribut / Rolle	Beschreibung	Beispielhafte Anwendung
Struktur und Navigation	role="navigation"	Kennzeichnet einen Bereich mit Navigationslinks.	Hauptmenü, Footer-Navigation
Struktur und Navigation	role="banner"	Oberer Seitenbereich, meist mit Logo oder Titel.	Kopfzeile einer Website
Struktur und Navigation	role="main"	Hauptinhalt der Seite, einzigartig pro Seite.	Inhaltsbereich eines Artikels
Struktur und Navigation	role="contentinfo"	Fußbereich mit Impressum, Kontakt etc.	Seitenfooter
Struktur und Navigation	role="region" + aria-label	Beschrifteter Inhaltsabschnitt.	„Ergebnisse der Suche“
Widgets und Interaktion	role="button"	Kennzeichnet klickbares Steuerelement.	Custom-Button
Widgets und Interaktion	role="dialog" + aria-modal="true"	Modales Fenster; blockiert Hintergrund.	Login-Dialog
Widgets und Interaktion	role="tablist", role="tab", role="tabpanel"	Struktur für Reiter-Navigation.	Registerkarten
Widgets und Interaktion	role="menu", role="menuitem"	Menü oder Kontextmenü.	Dropdown in App
Widgets und Interaktion	role="checkbox", aria-checked	Kontrollkästchen mit Zustand.	„Ich stimme zu“

Widgets und Interaktion	role="switch"	Umschalter zwischen zwei Zuständen.	Dark-Mode-Schalter
Status und Hinweise	role="alert"	Sofortige, wichtige Meldung.	Fehlermeldung bei Formular
Status und Hinweise	role="status"	Informative Änderung ohne Fokusänderung.	„Speichern abgeschlossen“
Status und Hinweise	aria-live="polite" / "assertive"	Bereich mit dynamischen Inhalten.	Chatnachrichten, Ladezustände
Beschriftung und Beziehungen	aria-label	Unsichtbare Kurzbeschreibung.	Symbol-Button ohne Text
Beschriftung und Beziehungen	aria-labelledby	Verknüpft mit sichtbarem Label.	Überschrift beschreibt Widget
Beschriftung und Beziehungen	aria-describedby	Verweist auf erläuternden Text.	Hilfetext für Eingabefeld
Beschriftung und Beziehungen	aria-details	Verknüpft ergänzende Informationen.	Verweis auf lange Beschreibung
Zustände und Eigenschaften	aria-expanded	Zeigt an, ob ein Bereich geöffnet ist.	Akkordeon
Zustände und Eigenschaften	aria-hidden	Verbirgt Element vor Screenreader.	Unsichtbare Inhalte
Zustände und Eigenschaften	aria-disabled	Element ist deaktiviert, aber sichtbar.	Inaktive Schaltfläche
Zustände und Eigenschaften	aria-required	Feld muss ausgefüllt werden.	Pflichtfeld in Formular
Zustände und Eigenschaften	aria-invalid	Markiert fehlerhafte Eingabe.	„E-Mail ungültig“
Zustände und Eigenschaften	aria-current	Aktives oder ausgewähltes Element.	Aktueller Menüpunkt

Checkliste zum Einsatz von ARIA Attributen

Grundprinzipien

- Weniger ist mehr: ARIA nur einsetzen, wenn HTML keine passende Lösung bietet.
- Semantik testen: Screenreader-Ausgabe prüfen – stimmt sie mit der visuellen Darstellung überein?
- Pflege sicherstellen: Bei jeder dynamischen Änderung (Ein- oder Ausblenden, Öffnen, Umschalten) müssen ARIA-Zustände aktualisiert werden.
- Dokumentation führen: Festlegen, welche ARIA-Attribute im Projekt erlaubt oder untersagt sind und warum.

Erweiterte Prüfpunkte

- Richtige Rollen verwenden: Prüfen, ob die zugewiesene role-Eigenschaft der tatsächlichen Funktion entspricht (zum Beispiel role="dialog" nur für echte Dialoge).
- Zustände korrekt setzen: Attribute wie aria-expanded, aria-pressed oder aria-selected müssen den aktuellen UI-Zustand widerspiegeln.
- Beziehungen prüfen: Wenn aria-labelledby oder aria-describedby verwendet wird, müssen die referenzierten IDs existieren und eindeutig sein.
- ARIA und Fokus gemeinsam denken: Ein Element mit role="dialog" oder role="alertdialog" muss beim Öffnen den Tastaturfokus korrekt setzen.

- Verstecken mit Bedacht: aria-hidden="true" darf nicht auf sichtbare, relevante Inhalte angewendet werden. Sichtbar bedeutet nicht automatisch zugänglich.
- Konsistenz über Komponenten sicherstellen: Gleiche Komponenten (zum Beispiel Akkordeons oder Tabs) müssen ARIA-Attribute konsistent verwenden.
- Frameworks prüfen: Überprüfen, welche ARIA-Werte von Frameworks wie React, Angular oder Vue automatisch gesetzt werden und welche ergänzt werden müssen.
- Test mit Assistive Technologien: Die Umsetzung immer mit mindestens einem Screenreader und Tastatur testen, um vorhersehbares Verhalten sicherzustellen.
- Keine widersprüchliche Semantik: Native HTML-Rollen dürfen nicht durch role überschrieben werden (zum Beispiel kein <button role="link">).
- ARIA und Live-Regionen prüfen: aria-live-Bereiche gezielt einsetzen und sicherstellen, dass Statusmeldungen tatsächlich vorgelesen werden.

Weitere Informationen

- Offizielle Spezifikation: W3C Recommendation – Accessible Rich Internet Applications (WAI-ARIA) 1.2
- Begleitdokument: ARIA Authoring Practices Guide (APG)
- Begleitdokument: ARIA in HTML
- Begleitdokument: Using ARIA
- Begleitdokument: Techniques for WCAG 2.2