Engineer

ARIA

Zusatzmarkup mit Bedienelementnamen für assistive Technik

Was es ist

ARIA ist eine W3C-Spezifikation für HTML-Attribute, die assistiver Technik Art, Namen und Zustand eines Seitenelements mitteilen, wenn HTML allein zu wenig sagt. Die Buchstaben stehen für Accessible Rich Internet Applications. Der vollständige Titel WAI-ARIA ergänzt die Web Accessibility Initiative des W3C. Die aktuelle Empfehlung ist Version 1.2.

Das Attribut `role` bezeichnet die Art eines Elements: ein Widget wie `button` oder `tab`, einen Strukturteil wie `list`, eine der acht Landmark-Rollen wie `navigation` oder `main` oder eine Live-Region wie `status`, deren Änderungen ein Screenreader ungefragt ansagt. Die `aria-*`-Attribute zerfallen in Eigenschaften wie `aria-labelledby` und Zustände wie `aria-expanded`, die sich mit den Handlungen des Nutzers ändern. Der Browser überträgt dies in die Barrierefreiheits-API des Betriebssystems. Diese liest ein Screenreader oder eine Sprachsteuerung.

ARIA verändert die Ansage und sonst nichts. Ein `div` mit `role="button"` wird als Schaltfläche vorgelesen, erhält aber keinen Fokus und ignoriert Enter und die Leertaste, bis `tabindex` und ein Skript beides ergänzen. Deshalb lautet die erste Regel im W3C-Leitfaden Using ARIA: Bietet ein natives HTML-Element oder -Attribut bereits die benötigte Semantik und das Verhalten, verwenden Sie es, statt ein anderes Element umzuwidmen und ARIA hinzuzufügen.

```
<!-- announced as a button, behaves like a div -->
<div role="button">Send</div>

<!-- announced, focusable and operable as a button -->
<button type="button">Send</button>
```

Die andere Hälfte ist der zugängliche Name: die Wörter, die ein Screenreader für ein Bedienelement spricht. Accessible Name and Description Computation 1.2 bestimmt die Reihenfolge: zuerst `aria-labelledby`, dann `aria-label`, danach die native Beschriftung wie ein `label`-Element, `alt` oder ein SVG-`title`, anschließend bei Rollen, die ihren Namen daraus beziehen, der Textinhalt und zuletzt das `title`-Attribut. Ein `aria-label` ergänzt die sichtbaren Wörter einer Schaltfläche daher nicht, sondern ersetzt sie.

Das Konzept für eine Entscheidung nutzen

Nutzen Sie „Was ist ARIA?“ für eine konkrete Entscheidung. Wir können helfen, das Konzept mit dem System in Verbindung zu bringen, das Sie entwickeln oder einkaufen. Oberflächen, die schnell laden, auf dem Smartphone funktionieren und sich in allen Browsern Ihrer Kunden gleich verhalten.

Unsere Leistungen entdecken Kontakt aufnehmen →

Die Annahmen hinter dem Begriff prüfen

Nutzen Sie die Erklärung und die verlinkten Arbeiten, um Fragen für Ihr Projekt zu formulieren. Wir können die Annahmen gemeinsam mit Ihnen prüfen, bevor ein Werkzeug ausgewählt wird. Full-Stack-GIS-Kartenentwicklung geleitet — Full-Stack-Kartenentwicklung von der Datenbank bis zum Browser

Full-Stack-GIS-Kartenentwicklung geleitet Kontakt aufnehmen →

Warum es wichtig ist

Die ARIA-Entscheidung eines Teams lautet Eigenbau oder Wiederverwendung. WCAG 2.2, Erfolgskriterium 4.1.2 Name, Rolle, Wert, verlangt, dass jede Oberflächenkomponente Namen und Rolle für Software bereitstellt, vom Nutzer einstellbare Zustände und Werte programmatisch setzen lässt und Änderungen meldet. Ein natives `button`, `select` oder `details` erfüllt dies von sich aus. Eine eigene Tab-Leiste, Combobox oder Baumansicht schafft es nur mit ARIA und dem Skript, das dessen Angaben korrekt hält.

Ein eigenes Widget verursacht deshalb wiederkehrende Kosten: Tastaturbedienung, Fokusverwaltung, Zustände für jede Interaktion und Tests in mehreren Screenreadern bei jeder Neugestaltung. Beurteilen Sie eine Komponentenbibliothek danach, welchem Muster des W3C ARIA Authoring Practices Guide jedes Widget folgt und ob dessen Tastaturverhalten ohne Maus tatsächlich vorhanden ist.

Der rechtliche Druck erreicht inzwischen private Unternehmen. Die Web Accessibility Directive, (EU) 2016/2102, betrifft öffentliche Websites und Apps. Der European Accessibility Act, Richtlinie (EU) 2019/882, gilt seit dem 28. Juni 2025 für verschiedene Verbraucherprodukte und -dienste, darunter E-Commerce und Privatkundenbankgeschäfte. Dienstleistungserbringende

Kleinstunternehmen sind ausgenommen. Beide stützen sich auf die harmonisierte Norm EN 301 549, deren Webanforderungen WCAG Level AA entsprechen. Name, Rolle, Wert gehört zu Level A. Bleiben Name oder Rolle einem Screenreader verborgen, scheitert das Element bereits an der niedrigsten geprüften Stufe.

Wie es in der Praxis aussieht

Die Vorlagen dieser Website verwenden ARIA sparsam. Die obere Leiste jeder Unterseite ist ein `div` mit `role="banner"` statt des üblichen `header` . Der Vorlagenkommentar begründet das: Das Stylesheet gestaltet ein unqualifiziertes `header` als bildschirmhohen Startseiten-Hero. Dieses Element würde die Geometrie auf jede Unterseite tragen; die Rolle liefert nur den Orientierungspunkt. Sie ordnet die Bildmarke als Startseitenlink einem Landmark zu. Das ist der enge Fall, den die erste Regel erlaubt. Das Menü geht den umgekehrten Weg: Ein Paar aus `details` und `summary` überlässt Öffnungszustand, Schaltflächensemantik und `aria-expanded` dem Browser. Das `nav` des Menüs und das `nav` der Fußzeile tragen unterschiedliche `aria-label` -Werte. Zwei gleichnamige Navigations-Landmarks verletzen die Eindeutigkeitsregel der Engine, und die Barrierefreiheitsprüfung der Website setzt null Befunde voraus. Die aufbewahrte Portfoliofilter-Implementierung verwendet eine `role="status"` -Region, die bis zur Eingabe still bleiben soll. Der Filter ist derzeit mit `searchEnabled = false` deaktiviert. Er veranschaulicht die aufbewahrte Implementierung, keine auf veröffentlichten Seiten verfügbare Interaktion.

Der Bericht darüber, die Website auf WCAG 2.2 Level AA zu bringen, liefert die andere Hälfte der Banner-Geschichte. Nach dem Einschalten der Best-Practice-Regeln scheiterte sofort eine davon, weil die Bildmarke auf jeder Unterseite zu keinem Landmark gehörte. Dieselbe Seite nennt unter ihren offengelegten Grenzen auch fehlende Tests mit echter assistiver Technik.

Die Beschreibung von elf Fehlern, die der Browser für unbedenklich hielt, zeigt die Entfernung zwischen Markup und Erfahrung. Alle elf lagen in der Darstellung, nicht in den Deklarationen. Nur Letztere verändert ARIA. Acht Fußzeilenlinks blieben per Tab erreichbar, nachdem ein Stil nur ihre Zeigerinteraktion abgeschaltet hatte. Es ist dieselbe Form wie `aria-hidden` auf einem fokussierbaren Element: Der Screenreader verliert es, die Tastatur behält es. Außerdem waren nur siebzig von hundertfünf Regeln aktiv. Die Regel für fehlende Landmarks gehörte zu den ausgeschalteten.

Die letzte Seite zeigt dasselbe Problem auf einer Plattform ohne ARIA. Alle siebzehn Oberflächensymbole für Screenreader benennen beschreibt eine SwiftUI-Anwendung ohne HTML und ohne ein einziges `aria-` -Attribut. Ihre Beschriftung ist SwiftUIs eigenes natives Gegenstück zu `aria-label` , nicht dessen Verwendung. Systemsymbole wurden unter ihrem internen Namen vorgelesen. Die Sendeschaltfläche meldete „Pfeil nach oben im Kreis, Schaltfläche“, und siebzehn reine Symbolsteuerungen beschrieben ihr Bild statt ihrer Funktion. Jede erhielt eine handlungsbezogene Beschriftung, die bei Bedarf dem Zustand folgt. Ein Linter verlangt nun innerhalb von acht Zeilen der erzeugenden Konstruktion eine Beschriftung. Ein achtzehntes Element kommt nur mit ihr hinein. Die genannte Grenze gilt ebenso für `aria-label` : Die Prüfung bestätigt nur irgendeine Beschriftung in der Nähe. Ob sie die richtige ist, kann nur ein zuhörender Mensch entscheiden.

Wo es schiefgeht

Eine Rolle sagt ein Widget an, das die Tastatur nicht erreicht. Ein `div` bekommt `role="button"` und einen Klickhandler. Der Screenreader sagt „Schaltfläche“, doch Tab überspringt es und Enter bleibt wirkungslos. Die dritte ARIA-Regel verlangt Tastaturbedienung für jedes interaktive ARIA-Element. Verwenden Sie das native Element oder implementieren Sie das vollständige Tastaturmodell des APG-Musters und testen mit abgezogener Maus.

Der Fokus landet auf etwas, das der Screenreader als nicht vorhanden behandelt. `aria-hidden="true"` steht auf einem fokussierbaren Element oder einem Vorfahren, etwa einer außerhalb des sichtbaren Bereichs liegenden Seitenleiste. Tab gelangt weiter dorthin, der Nutzer hört nichts. Die vierte ARIA-Regel verbietet das. Entfernen Sie verborgene Inhalte mit `hidden` , `display: none` oder `inert` auch aus der Tab-Reihenfolge. `aria-hidden` bleibt Dekoration vorbehalten, die nie Fokus erhält.

Ein Name steht an einem Element, das keinen annimmt. `aria-label` soll ein `div` , `span` oder `p` erklären, doch WAI-ARIA 1.2 verbietet Namen für `generic` und `paragraph` . Screenreader ignorieren sie oder lesen uneinheitlich. An einem Bedienelement ersetzt das Label die sichtbaren Wörter. Weichen beide ab, verfehlt ein Nutzer der Sprachsteuerung sein Ziel, wenn er den sichtbaren Text ausspricht. Benennen Sie nur Elemente, die Namen tragen können, und beginnen Sie jedes `aria-label` mit den sichtbaren Wörtern, wie WCAG 2.5.3 Beschriftung im Namen verlangt.

Zwei Rollen widersprechen einander. Eine ausdrückliche Rolle ersetzt die native. Ein Link mit `role="button"` wird als Schaltfläche angesagt, scrollt aber mit der Leertaste. `role="tab"` auf `h2` entfernt die Überschrift aus der Überschriftenliste, `role="navigation"` auf `ul` verliert Liste und Elementzahl. Die zweite Regel erlaubt Änderungen nativer Semantik nur, wenn sie notwendig sind. ARIA in HTML führt die erlaubten Rollen pro Element auf.

Eine Live-Region redet ununterbrochen. `aria-live="assertive"` oder `role="alert"` sitzt an einer Ergebniszahl oder einem Countdown, die sich bei jedem Tastendruck oder jede Sekunde ändern. Der Screenreader unterbricht für jeden Wert. Verwenden Sie zurückhaltende (polite) Aktualisierungen, wie sie `role="status"` bereits definiert. Melden Sie einmal nach dem Ende der Eingabe und lassen Sie die Region beim Laden still.

Weiterführende Lektüre

- WAI-ARIA 1.2 — Rollen, Zustände und Eigenschaften, erlaubte Attribute und Rollen ohne zulässigen Namen.
- ARIA in HTML — erlaubte Rollen und Attribute je HTML-Element sowie redundante Angaben.
- Using ARIA — der praktische W3C-Leitfaden mit Nutzungsregeln und einem Beispiel zu jeder Regel.
- Accessible Name and Description Computation 1.2 — der Algorithmus für den vom Screenreader gesprochenen Namen.
- Erläuterung zu Name, Rolle, Wert — WCAG 2.2, Kriterium 4.1.2, mit Techniken und Fehlerfällen für jede Komponente.
- ARIA Authoring Practices Guide — Markup und Tastaturverhalten gängiger Widgetmuster mit funktionierenden Beispielen.

Glossar

Für Ihre tatsächlichen Rahmenbedingungen auswählen

Beschreiben Sie den Arbeitsablauf, die vorhandenen Schnittstellen und die Nutzer, die die Veränderung benötigen. Wir können eine prüfbare Lieferung definieren und die Übergabe so gestalten, dass Ihr Team die gelieferten Bestandteile warten kann.

Preise ansehen Kontakt aufnehmen →

Die Begriffe praktisch nutzbar machen

Fragen Sie, wie „Was ist ARIA?“ auf Ihre Organisation zutrifft. Ein konkretes Beispiel aus Ihrem bestehenden System reicht aus, um das Gespräch zu beginnen.

Unsere Leistungen entdecken

welcome@engineer.company +45 50 10 61 84 Kopenhagen, Dänemark

Nehmen Sie diese Seite mit

Kontaktkarte

Mit Sorgfalt und Liebe gestaltet von D.

CVR 45969703 · Tagensvej 105C, 3. th, 2200 København N

Copyright © 2025 – heute · Engineer ApS · Alle Rechte vorbehalten.

Gedruckt von https://engineer.company/de/glossary/aria/