**Engineer**

# ARIA

ekstra markering som navngir en kontroll for hjelpemidler

## Hva det er

ARIA, med fullt navn WAI-ARIA, er en W3C-spesifikasjon for HTML-attributter som forteller hjelpemiddelteknologi hva et element på en nettside er, hva det heter og hvilken tilstand det er i, der HTML alene sier for lite. Bokstavene står for Accessible Rich Internet Applications, og WAI i det fulle navnet viser til Web Accessibility Initiative i W3C. Gjeldende anbefaling er versjon 1.2.

Attributtet `role` sier hva slags ting elementet er: en kontroll som `button` eller `tab`, en del av strukturen som `list`, en av de åtte rollene for landemerker som `navigation` eller `main`, eller en live-region som `status`, der skjermleseren leser opp endringene uten at noen ber om det. `aria-*` -attributtene deler seg deretter i egenskaper, som `aria-labelledby`, og tilstander, som `aria-expanded`, som skifter mens brukeren arbeider. Nettleseren oversetter alt sammen til operativsystemets tilgjengelighets-API, og det er der en skjermleser eller et program for stemmestyring henter opplysningene sine.

ARIA endrer hva som blir lest opp, og bare det. En `div` med `role="button"` leses opp som en skjermknapp, men den får ikke fokus, og Enter og mellomrom gjør ingenting før et `tabindex` og et skript sørger for begge deler. Derfor lyder den første regelen for bruk av ARIA i W3C-veiledningen Using ARIA omtrent slik: Når et innebygd HTML-element eller attributt allerede har den semantikken og oppførselen du trenger, bruker du det, i stedet for å gi et annet element ny mening og legge ARIA oppå.

```
<!-- leses opp som en knapp, oppfører seg som en div -->
<div role="button">Send</div>

<!-- leses opp, får fokus og betjenes som en knapp -->
<button type="button">Send</button>
```

Den andre halvdelen er det tilgjengelige navnet, ordene en skjermleser sier for en kontroll. Accessible Name and Description Computation 1.2 fastsetter rekkefølgen: først `aria-labelledby`, så `aria-label`, så vertsspråkets egen etikett (et `label`-element, en `alt`, en `title` i SVG), deretter elementets tekstinnhold for roller som henter navnet sitt derfra, og til slutt attributtet `title`. Et `aria-label` på en knapp erstatter dermed de synlige ordene i stedet for å legge noe til dem.

Bruk begrepet i en beslutning

Bruk «Hva er ARIA?» i en konkret beslutning. Vi kan knytte begrepet til systemet dere bygger eller kjøper. Grensesnitt som laster raskt, fungerer på mobilen og oppfører seg likt i alle nettlesere kundene dine måtte bruke.

**Se tjenestene våre** Ta kontakt →

Undersøk antakelsene bak ordet

Bruk forklaringen og det tilknyttede arbeidet til å formulere spørsmål for prosjektet. Vi kan undersøke antakelsene sammen før et verktøy velges. Styrte fullstack-utviklingen av GIS-kart — fullstack-kartutvikling, fra database til nettleser

**Styrte fullstack-utviklingen av GIS-kart** Ta kontakt →

## Hvorfor det er viktig

Valget ARIA legger foran et team, er om det skal bygge selv eller bruke det som allerede finnes. Suksesskriterium 4.1.2 i WCAG 2.2, Name, Role, Value, krever at hver komponent i et brukergrensesnitt gjør navnet og rollen sin tilgjengelig for programvare, lar tilstander og verdier som brukeren kan endre, settes programmatisk, og melder fra når de endres. En innebygd `button`, `select` eller `details` oppfyller dette uten videre. En hjemmelaget fanerad, kombinasjonsboks eller trestruktur oppfyller det bare gjennom ARIA og skriptet som holder ARIA-verdiene sanne.

Dermed blir en hjemmelaget kontroll en løpende kostnad: tastaturhåndtering, fokusstyring, tilstander som følger hver handling, og testing i mer enn én skjermleser ved hver redesign. Vurder et komponentbibliotek etter hvilket mønster fra ARIA Authoring Practices Guide (APG) hver kontroll følger, og etter om mønsterets tastaturoppførsel er på plass når musen er lagt bort.

Det rettslige presset når nå også private virksomheter. Webdirektivet, (EU) 2016/2102, dekker nettsteder og apper i offentlig sektor. Tilgjengelighetsdirektivet, direktiv (EU) 2019/882, kjent som European Accessibility Act (EAA), gjelder i EU fra 28. juni 2025 for en rekke produkter og tjenester til forbrukere, bl.a. netthandel og banktjenester, mens mikroforetak som leverer tjenester, er unntatt. Begge lener seg på den harmoniserte

standarden EN 301 549, der kravene til web er WCAG på nivå AA. Name, Role, Value ligger på nivå A, så en kontroll der navnet eller rollen aldri når frem til skjermleseren, stryker allerede på det laveste nivået en revisjon sjekker.

I Norge skal ikt-løsninger være universelt utformet, og ifølge Uutilsynet gjelder kravet både private og offentlige virksomheter. Endringene som fulgte av webdirektivet, gjelder fra 1. februar 2023. Tilgjengelighetsdirektivet trådte i kraft i EU 28. juni 2025, men i EØS-landene er gjennomføringen forsinket, og Norge er i ferd med å innlemme det i nasjonalt regelverk.

## Slik ser det ut i praksis

Malene bak dette nettstedet bruker ARIA sparsomt. Topplinjen på hver underside er en `div` med `role="banner"` og ikke det vanlige `header`-elementet. Kommentaren i malen forklarer hvorfor: stilarket gjør et nakent `header`-element til forsidens store toppfelt i full høyde, så elementet ville tatt med seg den geometrien til hver underside, mens rollen gir landemerket og ingenting mer. Det er rollen som plasserer merkevaresymbolet, lenken hjem, inne i et landemerke, og det er nettopp det smale tilfellet den første regelen åpner for. Menyen går motsatt vei: den er et par av `details` og `summary`, som overlater åpen eller lukket tilstand, knappesemantikken og `aria-expanded` til nettleseren. Menyens `nav` og bunntekstens `nav` har hver sin `aria-label`, fordi to navigasjonslandemerker med samme navn bryter regelen om entydige navn i regelmotoren, og tilgjengelighetssjekken på nettstedet slipper bare gjennom null funn. Filterimplementasjonen som er beholdt i kildekoden, bruker en region med `role="status"` som er laget for å holde seg stille til leseren skriver. Filteret er for tiden deaktivert med `searchEnabled = false`; det viser den beholdte implementasjonen, ikke en interaksjon som er tilgjengelig på de publiserte sidene.

Siden om hvordan nettstedet nådde WCAG 2.2 nivå AA gir den andre halvdelen av historien om topplinjen. Da reglene for beste praksis ble slått på i en regelmotor for universell utforming, feilet én av dem med en gang, fordi merkevaresymbolet på hver underside lå utenfor alle landemerker. Samme side oppgir blant begrensningene sine at det ikke er testet med ekte hjelpemiddelteknologi.

Gjennomgangen av elleve feil som nettleseren kalte friske, viser hvor langt oppmerkingen står fra opplevelsen. Alle elleve lå i hvordan siden ble gjengitt, og ikke i hva oppmerkingen sa, og oppmerkingen er det eneste ARIA berører. Åtte

lenker i bunnteksten ble værende i tabulatorrekkefølgen etter at en stilregel hadde slått av pekerinteraksjon med dem og latt alt annet være: samme mønster som `aria-hidden` på et element som kan få fokus, der skjermleseren mister elementet mens tastaturet beholder det. Dessuten var bare sytti av regelmotorens hundre og fem regler slått på, og regelen som fanger et manglende landemerke, var blant dem som sto av.

Den siste siden viser det samme problemet på en plattform der ARIA ikke spiller noen rolle. Siden om de sytten ikonene som fikk navn for skjermlesere, handler om en SwiftUI-app uten HTML og uten et eneste `aria-` -attributt. Tilgjengelighetsetiketten der hører til SwiftUI selv, som det innebygde motstykket til `aria-label` og ikke en variant av det. Kontroller tegnet med systemsymboler ble lest opp med symbolets interne navn, så sendeknappen ble lest opp som «arrow up circle, button», og sytten rene ikonkontroller beskrev grafikken sin i stedet for funksjonen. Hver av dem fikk en etikett som navngir handlingen og følger tilstanden der det betyr noe, og en linter krever nå en etikett innenfor åtte linjer fra konstruksjonen som lager en ren ikonkontroll, så en attende kontroll slipper bare inn med etikett. Siden sier selv hvor grensen går, og den gjelder like mye for `aria-label` : sjekken bekrefter at det finnes en etikett i nærheten, mens bare et menneske som lytter, kan avgjøre om den er den riktige.

## Hvor det går galt

**En rolle lover en kontroll som tastaturet aldri når frem til.** En `div` får `role="button"` og en klikkhandler. Skjermleseren sier «knapp», men Tab hopper over den, og Enter gjør ingenting. Den tredje regelen for bruk av ARIA krever at alle interaktive ARIA-kontroller kan betjenes med tastaturet. Bruk det innebygde elementet, eller bygg hele tastaturmodellen fra APG-mønsteret og test med musen koblet fra.

**Fokus lander på noe skjermleseren regner som fraværende.** `aria-hidden="true"` sitter på et element som kan få fokus, eller på et overordnet element, f.eks. en skuffemeny utenfor skjermen. Tab fører fortsatt dit, og brukeren hører stillhet. Den fjerde regelen for bruk av ARIA utelukker nettopp dette. Ta skjult innhold ut av tabulatorrekkefølgen også, med `hidden` , `display: none` eller `inert` , og spar `aria-hidden` til pynt som aldri får fokus.

**Et navn havner på et element som avviser navn.** Et `aria-label` på en `div`, `span` eller `p` er ment som en forklaring, men WAI-ARIA 1.2 forbyr å gi rollene `generic` og `paragraph` et navn, og skjermlesere overser det eller leser det opp ujevnt. På en kontroll erstatter etiketten de synlige ordene, og når de to er forskjellige, bommer en bruker av stemmestyring som sier det som står på skjermen. Gi navn til elementer som kan bære et, og la hvert `aria-label` begynne med de synlige ordene, slik WCAG 2.5.3, Label in Name, ber om.

**To roller sier imot hverandre.** En eksplisitt rolle erstatter elementets egen. En lenke med `role="button"` leses opp som en knapp, men ruller siden ved trykk på mellomrom; `role="tab"` på en `h2` fjerner den fra listen over overskrifter; `role="navigation"` på en `ul` visker ut listen og antallet punkter. Den andre regelen for bruk av ARIA er å endre innebygd semantikk bare når du må, og ARIA in HTML lister opp hvilke roller hvert element kan få.

**En live-region snakker i ett kjør.** `aria-live="assertive"` eller `role="alert"` sitter på et antall treff eller en nedtelling som endres ved hvert tastetrykk eller hvert sekund, og skjermleseren avbryter brukeren for hver nye verdi. Bruk høflige oppdateringer, som `role="status"` gir per definisjon, meld fra først når inntastingen har roet seg, og hold regionen stille når siden lastes.

## Les mer

- WAI-ARIA 1.2 — alle roller, tilstander og egenskaper, hvilke attributter hver rolle støtter, og hvilke roller som ikke kan få navn.
- ARIA in HTML — rollene og attributtene hvert HTML-element kan ha, og hvilke av dem som er overflødige.
- Using ARIA — den praktiske veiledningen fra W3C, med reglene for bruk av ARIA og et eksempel på hver.
- Accessible Name and Description Computation 1.2 — algoritmen som avgjør hva en skjermleser sier at en kontroll heter.
- Understanding Name, Role, Value — hva suksesskriterium 4.1.2 i WCAG 2.2 krever av hver komponent, med teknikker og typiske feil.
- ARIA Authoring Practices Guide — oppmerking og tastaturoppførsel for hvert vanlige kontrollmønster, med eksempler som virker.



Ordliste

Velg ut fra deres faktiske rammer

Beskriv arbeidsflyten, eksisterende grensesnitt og brukerne som trenger endringen. Vi kan definere en testbar leveranse og en overlevering som gjør teamet i stand til å vedlikeholde løsningen.

**Se hva det koster** Ta kontakt →

# Gjør begrepene praktiske

Spør hvordan «Hva er ARIA?» gjelder i organisasjonen deres. Et konkret eksempel fra dagens system er nok til å starte samtalen.

**Se tjenestene våre**

welcome@engineer.company +45 50 10 61 84 København, Danmark

Ta med deg denne siden

Visittkort

**Laget med kjærlighet av D.**

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

**Opphavsrett © 2025 – nå · Engineer ApS · Alle rettigheter forbeholdt.**

**Skrevet ut fra https://engineer.company/nb/glossary/aria/**