Engineer

ARIA

розмітка, що називає елемент для допоміжних технологій

Що це таке

ARIA — це специфікація W3C, набір атрибутів HTML, які повідомляють допоміжним технологіям, чим є елемент сторінки, як він називається і в якому стані перебуває, коли самого HTML для цього замало. Абревіатура розшифровується як Accessible Rich Internet Applications, а повна назва специфікації, WAI-ARIA, додає до неї Web Accessibility Initiative, ініціативу W3C з вебдоступності. Чинна рекомендація W3C має версію 1.2.

Атрибут `role` каже, що це за річ: віджет на кшталт `button` чи `tab`, структурна частина на кшталт `list`, одна з восьми ролей-орієнтирів, як-от `navigation` або `main`, чи жива область (live region) на кшталт `status`, зміни в якій зчитувач екрана озвучує сам, без запиту. Решта атрибутів `aria-*` поділяється на властивості, наприклад `aria-labelledby`, і стани, наприклад `aria-expanded`, які змінюються разом із діями користувача. Браузер переносить усе це в API доступності операційної системи, а вже звідти інформацію бере зчитувач екрана чи програма голосового керування.

ARIA змінює лише те, що буде озвучено. Зчитувач назве `div` з `role="button"` кнопкою, проте фокус на такий елемент не потрапляє, а Enter і пробіл нічого не роблять, доки `tabindex` і скрипт не додадуть одне й друге. Тому перше правило використання ARIA в посібнику W3C Using ARIA звучить так: якщо вже існує вбудований елемент чи атрибут HTML із потрібними семантикою та поведінкою, використовуйте його, а не перепрофільовуйте інший елемент, додаючи до нього ARIA.

```
<!-- озвучується як кнопка, поводиться як div -->
<div role="button">Надіслати</div>

<!-- озвучується, отримує фокус і працює як кнопка -->
```

```
<button type="button">Надіслати</button>
```

Друга половина — доступна назва, тобто слова, якими зчитувач екрана називає елемент керування. Специфікація Accessible Name and Description Computation 1.2 задає порядок: спершу `aria-labelledby`, далі `aria-label`, потім власна мітка мови розмітки (елемент `label`, атрибут `alt`, `title` у SVG), потім текстовий вміст елемента для ролей, які беруть назву з нього, і наостанок атрибут `title`. Отже, `aria-label` на кнопці замінює видимі слова, а не доповнює їх.

Застосуйте поняття до рішення

Застосуйте «Що таке ARIA?» до конкретного рішення. Ми можемо пов'язати поняття із системою, яку ви створюєте або купуєте. Інтерфейси, що швидко завантажуються, працюють на телефоні й поводяться однаково в усіх браузерах ваших клієнтів.

Переглянути наші послуги Зв'язатися з нами →

Перевірте припущення за терміном

Використайте пояснення та пов'язані роботи, щоб сформулювати запитання для проєкту. Ми можемо перевірити припущення разом до вибору інструмента. Керувала full-stack-розробкою GIS-карт — full-stack-картографія — від бази даних до браузера

Керувала full-stack-розробкою GIS-карт Зв'язатися з нами →

Чому це важливо

ARIA ставить перед командою вибір: будувати самим чи взяти готове. Критерій успішності 4.1.2 у WCAG 2.2, Name, Role, Value, вимагає, щоб кожен компонент інтерфейсу відкривав програмам свою назву й роль, дозволяв програмно встановлювати стани та значення, які змінює користувач, і повідомляв про їхні зміни. Вбудовані `button`, `select` чи `details` виконують цю вимогу одразу. Власна смуга вкладок, комбінований список чи дерево виконують її лише завдяки ARIA і скрипту, який тримає атрибути в згоді з дійсністю.

Тож власний віджет — це постійні витрати: обробка клавіатури, керування фокусом, стани, що відстежують кожну дію, і тестування з кількома зчитувачами екрана після кожного редизайну. Оцінюючи бібліотеку компонентів, з’ясуйте, за яким шаблоном з ARIA Authoring Practices Guide від W3C зроблено кожен віджет і чи працює клавіатурна поведінка цього шаблону, коли мишу відкладено.

Юридичний тиск тепер сягає й приватних компаній. Директива про доступність вебсайтів, (ЄС) 2016/2102, охоплює сайти й мобільні застосунки органів публічного сектору. Директива (ЄС) 2019/882, відома як European Accessibility Act, застосовується з 28 червня 2025 року до низки споживчих продуктів і послуг, серед них електронна комерція та банківські послуги для споживачів, а мікропідприємства, що надають послуги, звільнені від її вимог. Обидві спираються на гармонізований стандарт EN 301 549, чиї вимоги до вебу — це WCAG рівня AA. Name, Role, Value стоїть на рівні A, тож елемент керування, чия назва чи роль лишається прихованою від зчитувача екрана, провалює найнижчий рівень, який перевіряє аудитор.

Як це виглядає на практиці

Шаблони цього сайту вдаються до ARIA ощадливо. Верхня панель на кожній внутрішній сторінці — це `div` з `role="banner"`, а не звичний елемент `header`. Коментар у шаблоні пояснює чому: таблиця стилів оформлює голий `header` як великий вступний блок головної сторінки на всю висоту екрана, тож елемент переніс би цю геометрію на кожну внутрішню сторінку, а роль дає орієнтир і більше нічого. Саме роль вміщує знак бренду, тобто посилання на головну, всередину орієнтира. Це якраз той вузький випадок, який перше правило допускає. Меню сайту йде протилежним шляхом: це пара `details` і `summary`, яка лишає браузерові стан відкриття, семантику кнопки та `aria-expanded`. Елементи `nav` меню і футера мають різні значення `aria-label`, бо два навігаційні орієнтири з однією назвою порушують правило унікальності в рушії перевірок, а перевірка доступності сайту вимагає нуля зауважень. Збережена в коді реалізація фільтра портфоліо використовує область `role="status"`, яка за задумом мовчить, доки читач не почне вводити текст. Зараз цей фільтр вимкнено через `searchEnabled = false`; він ілюструє збережену реалізацію, а не взаємодію, доступну на опублікованих сторінках.

Опис того, як сайт довели до WCAG 2.2 рівня AA, дописує другу половину історії верхньої панелі: щойно в рушії перевірок увімкнули правила найкращих практик, одне з них одразу дало збій, бо на кожній внутрішній сторінці знак бренду лишався за межами всіх орієнтирів. Та сама сторінка серед обмежень, які вона публікує, називає відсутність тестування з реальними допоміжними технологіями.

Розбір одинадцяти дефектів, які браузер визнав справними, показує, як далеко розмітка від того, що відчуває користувач. Усі одинадцять жили в тому, як сторінка малювалася на екрані, а не в тому, що оголошувала розмітка, а розмітка — це все, чого торкається ARIA. Вісім посилань у футері

й далі були досяжні клавішею Tab, хоча стиль вимкнув для них взаємодію з вказівником і більше нічого не змінив. Це та сама схема, що й `aria-hidden` на елементі, здатному отримати фокус: зчитувач екрана елемент втрачає, а клавіатура досі до нього веде. До того ж у рушії було ввімкнено лише 70 правил зі 105, і правило, що мало б помітити відсутній орієнтир, опинилося серед вимкнених.

Остання сторінка показує ту саму проблему на платформі, де ARIA немає зовсім. Усі сімнадцять іконок інтерфейсу отримали назви для зчитувачів екрана — це історія застосунку на SwiftUI, де немає ні HTML, ні жодного атрибута `aria-` . Мітка доступності там належить самому SwiftUI: це вбудований відповідник `aria-label` , а не його різновид. Елементи керування, намальовані системними символами, зачитувалися внутрішньою назвою символу, тож кнопка надсилання звучала як «стрілка вгору коло, кнопка», а сімнадцять елементів керування з самою іконкою описували свій малюнок замість своєї функції. Кожен отримав мітку, що називає дію і там, де це важливо, стежить за станом, а лінтер тепер відхиляє елемент керування з іконкою, якщо мітки немає в межах восьми рядків від місця, де його створено, тож вісімнадцятий потрапить у застосунок лише з міткою. Сторінка сама називає межу, і вона стосується й `aria-label` : перевірка підтверджує, що якась мітка стоїть поблизу, а чи вона правильна, може вирішити лише людина, яка її слухає.

Де виникають проблеми

Роль обіцяє віджет, до якого клавіатура не дістає. Елемент `div` отримує `role="button"` і обробник кліку. Зчитувач екрана каже «кнопка», але Tab його проминає, а Enter нічого не робить. Третє правило використання ARIA вимагає, щоб кожен інтерактивний елемент керування з ARIA працював із клавіатури. Візьміть вбудований елемент або побудуйте всю клавіатурну модель шаблону з APG і тестуйте з від’єднаною мишею.

Фокус потрапляє на те, що зчитувач екрана вважає відсутнім. Атрибут `aria-hidden="true"` стоїть на елементі, здатному отримати фокус, або на його предкові, наприклад на висувній панелі за межами екрана. Tab і далі веде туди, а користувач чує тишу. Четверте правило використання ARIA прямо це виключає. Прибирайте прихований вміст і з порядку табуляції за допомогою `hidden` , `display: none` або `inert` , а `aria-hidden` лишайте для оздоблення, яке ніколи не отримує фокусу.

Назва потрапляє туди, де елемент її не приймає. Атрибут `aria-label` на `div`, `span` чи `p` мав би щось пояснити, але WAI-ARIA 1.2 забороняє давати назву ролям `generic` і `paragraph`, тож зчитувачі екрана її ігнорують або читають непослідовно. На елементі керування мітка замінює видимі слова, і коли вони розходяться, користувач голосового керування промахується, вимовляючи те, що бачить на екрані. Давайте назви лише елементам, які можуть її мати, і починайте кожен `aria-label` з видимих слів, як вимагає WCAG 2.5.3, Label in Name.

Дві ролі суперечать одна одній. Явна роль замінює власну роль елемента. Посилання з `role="button"` озвучується як кнопка, але пробіл на ньому прокручує сторінку; `role="tab"` на `h2` прибирає його зі списку заголовків; `role="navigation"` на `ul` позбавляє його ролі списку й кількості пунктів. Друге правило використання ARIA — змінювати вбудовану семантику лише тоді, коли без цього не обійтися, а ARIA in HTML перелічує, які ролі може мати кожен елемент.

Жива область говорить без упину. Атрибут `aria-live="assertive"` або `role="alert"` стоїть на лічильнику результатів чи зворотному відліку, що змінюється з кожним натисканням клавіші чи щосекунди, і зчитувач екрана перебиває користувача на кожному новому значенні. Використовуйте ввічливі оголошення (`polite`), які `role="status"` дає за визначенням, озвучуйте результат, коли введення вщухне, і тримайте область тихою під час завантаження сторінки.

Що почитати далі

- WAI-ARIA 1.2 — усі ролі, стани й властивості, які атрибути підтримує кожна роль і яким ролям не можна давати назву.
- ARIA in HTML — які ролі й атрибути може мати кожен елемент HTML і які з них зайві.
- Using ARIA — практичний посібник W3C із правилами використання ARIA і прикладом до кожного.
- Accessible Name and Description Computation 1.2 — алгоритм, що визначає, як зчитувач екрана назве елемент керування.
- Understanding Name, Role, Value — чого критерій успішності 4.1.2 у WCAG 2.2 вимагає від кожного компонента, з техніками й типовими помилками.
- ARIA Authoring Practices Guide — розмітка й клавіатурна поведінка для кожного поширеного шаблону віджета, з робочими прикладами.

Словник

Обирайте за фактичними умовами

Опишіть робочий процес, наявні інтерфейси та користувачів, яким потрібна зміна. Ми можемо визначити перевірюваний результат і спосіб передачі, який дасть вашій команді змогу підтримувати його.

Дивитися, скільки це коштує Зв'язатися з нами →

Зробіть термінологію практичною

Запитайте, як «Що таке ARIA?» стосується вашої організації. Конкретного прикладу з поточної системи достатньо для початку розмови.

Переглянути наші послуги

welcome@engineer.company +45 50 10 61 84 Копенгаген, Данія

Візьміть цю сторінку із собою

Візитівка

Дбайливо зроблено з любов'ю — D.

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

Авторське право © 2025 – дотепер · Engineer ApS · Усі права захищено.

Надруковано з https://engineer.company/uk/glossary/aria/