La voz de ASEPAU

ARIA es el nuevo amigo de la accesibilidad digital

Jonathan Chacón Barbero
Socio de ASEPAU
Consultor en accesibilidad, usabilidad y nuevas tecnologías

La evolución de la World Wide Web ha sido un proceso constante y acelerado que ha transformado radicalmente la manera en que las personas interactúan con la información y los servicios digitales. Al principio la web se identificaba como un repositorio de documentos estáticos enlazados entre sí, una estructura para la cual el lenguaje HTML proporcionaba la semántica suficiente y necesaria. Pero en los últimos años se ha producido un cambio hacia la web como plataforma de aplicaciones. Este tránsito desde el documento hacia la aplicación ha generado tensiones significativas en el campo de la accesibilidad digital, especialmente en lo que respecta al uso y la percepción de WAI-ARIA (Web Accessibility Initiatives - Accessible Rich Internet Applications).

¿Qué es ARIA?

WAI-ARIA, más conocida como ARIA, es una especificación del W3C que define una capa semántica adicional para mejorar la accesibilidad de interfaces web dinámicas, especialmente cuando el HTML nativo no describe suficientemente la función, el estado o la estructura de ciertos componentes. Utilizando diversos roles y atributos, ARIA permite que tecnologías de asistencia como lectores de pantalla interpreten correctamente widgets personalizados, regiones dinámicas y cambios de contenido en aplicaciones web.

ARIA puede resolver carencias de accesibilidad en componentes web creados sin ningún elemento semántico. Pero hay que proporcionar todos los atributos y roles de accesibilidad para ese componente de forma absoluta y apropiada.

ARIA no añade funcionalidad por sí misma: si se declara un rol interactivo, también se debe implementar teclado, foco, estados y eventos coherentes. Un ARIA mal aplicado puede degradar la accesibilidad, por lo que conviene seguir patrones probados y validar con pruebas —teclado, lector de pantalla y herramientas de auditoría—.

¿Por qué ARIA se identificaba como enemigo de la accesibilidad?

Mantener una postura de rechazo absoluto hacia ARIA no solo es anacrónico, sino que se ha convertido en una práctica peligrosa que amenaza con excluir a las personas con discapacidad de las experiencias digitales modernas.

Durante años, muchos consultores y auditores de accesibilidad han identificado a ARIA como un elemento hostil, un vector de problemas más que de soluciones; acuñando la máxima de que la mejor forma de usar ARIA es no usándola. Si bien esta postura pudo tener una justificación empírica y pragmática hace una década, el actual panorama tecnológico, dominado por las Aplicaciones Web Progresivas (PWA) y los frameworks de JavaScript, exige una revisión urgente y profunda de este dogma. Mantener una postura de rechazo absoluto hacia ARIA no solo es anacrónico, sino que se ha convertido en una práctica peligrosa que amenaza con excluir a las personas con discapacidad de las experiencias digitales modernas y futuras.

Históricamente, la reticencia hacia la implementación de ARIA se fundamentaba en la «Primera Regla de ARIA», la cual establece que si un elemento nativo de HTML puede lograr el comportamiento y la semántica deseados, debe preferirse sobre la reimplementación mediante ARIA y JavaScript. Esta regla, en su esencia, sigue siendo válida para documentos web tradicionales donde los elementos semánticos del HTML5 cubren la gran mayoría de las necesidades estructurales. No obstante, la interpretación dogmática de esta regla ha llevado a una demonización de la tecnología. Muchos profesionales, al observar la cantidad de errores introducidos por desarrolladores inexpertos al intentar enriquecer sus interfaces —como etiquetas mal aplicadas, estados incoherentes o roles redundantes—, optaron por la solución más drástica: desaconsejar su uso por completo. Esta aproximación reduccionista ignora la realidad de la ingeniería de software actual. La web de hoy no se construye página a página, sino componente a componente, utilizando librerías y marcos de trabajo como React, Angular, Vue o Svelte, los cuales manipulan el Modelo de Objetos del Documento (DOM) de manera dinámica y reactiva. En este contexto, JavaScript ha dejado de ser un complemento opcional o un enemigo de la accesibilidad para convertirse en el tejido conectivo indispensable de la web moderna.

Esta demonización de ARIA por parte de los profesionales de la accesibilidad provocó que los desarrolladores y diseñadores no se molestasen por conocer ARIA, cómo se utiliza de forma correcta o cómo se incorpora en los procesos de diseño y desarrollo de componentes web.

La situación actual

El problema que enfrentamos en la actualidad no es la tecnología ARIA en sí misma, sino la complejidad inherente a las interfaces que intentamos construir y la incapacidad del HTML estándar para modelarlas por sí solo. Las grandes empresas tecnológicas, como Google, Microsoft y Apple, están impulsando fuertemente el modelo de las WebApps, difuminando las fronteras entre las aplicaciones nativas de un sistema operativo y las aplicaciones que se ejecutan en un navegador. Este tipo de experiencia de usuario busca emular la interactividad, la fluidez y la complejidad de una aplicación de escritorio o móvil. Para lograr esto, los diseñadores y desarrolladores crean controles y mecanismos de navegación que no tienen un equivalente directo en el estándar HTML. Elementos como árboles de navegación expandibles, tablas de datos interactivas, menús combinados con autocompletado, pestañas dinámicas y ventanas modales complejas son patrones de diseño habituales que el HTML, por su naturaleza declarativa y orientada a documentos, no puede describir semánticamente de forma nativa.

La función primordial de ARIA es actuar como un puente semántico entre el DOM y el Árbol de Accesibilidad que los navegadores exponen a los productos de apoyo.

Es en este preciso punto de inflexión donde ARIA deja de ser un complemento opcional para convertirse en una infraestructura crítica. La función primordial de ARIA es actuar como un puente semántico entre el DOM y el Árbol de Accesibilidad que los navegadores exponen a los productos de apoyo, como los lectores de pantalla. Cuando una aplicación web moderna renderiza un componente visual complejo, el navegador puede dibujar los píxeles, pero sin ARIA, ese componente es semánticamente invisible o confuso para una persona ciega. ARIA permite a los desarrolladores comunicar al producto de apoyo no solo qué es un elemento —su rol—, sino también cuál es su estado actual —si está desplegado, seleccionado, presionado o deshabilitado— y cuáles son sus propiedades — si es requerido, si tiene un menú emergente asociado o si controla otra parte de la interfaz—. Negar el uso de ARIA en el desarrollo de WebApps equivale a negar a las personas usuarias de productos de apoyo el acceso a la información sobre el estado y la funcionalidad de la

Es indispensable comprender que la interacción con una aplicación web difiere cognitivamente y mecánicamente de la lectura de un documento web. Los lectores de pantalla operan tradicionalmente interceptando el DOM y permitiendo al usuario navegar linealmente o saltar entre estructuras semánticas mediante un cursor virtual. Sin embargo, para interactuar con una aplicación compleja, los usuarios a menudo necesitan que su producto de apoyo cambie de modo de comportamiento, pasando de un modo de lectura a un modo de interacción o formulario, donde las teclas pulsadas se envían directamente a la aplicación web en lugar de ser interceptadas por el lector para la navegación. ARIA es el mecanismo que señala este cambio de contexto. Sin los roles adecuados y la gestión de foco, un usuario podría quedar atrapado en un control personalizado sin saber cómo interactuar con él o cómo salir. Por esta razón, la ausencia de ARIA en estos entornos no resulta en una web más simple y accesible, sino en una barrera de accesibilidad difícil de superar.

El problema no es la herramienta, sino la falta de formación especializada en su uso.

La postura de evitar ARIA ignora también la evolución de los propios productos de apoyo y cómo estos se han adaptado para interpretar la nueva semántica que proviene de las aplicaciones modernas. Los motores de accesibilidad de los sistemas operativos y los navegadores han mejorado sustancialmente su soporte para la especificación ARIA, permitiendo experiencias de usuario que antes eran imposibles en la web. No obstante, para que esta orquestación tecnológica funcione, es necesario que la información suministrada por el código sea precisa. Aquí radica el verdadero desafío y el origen de la mala reputación de ARIA: la brecha de competencia técnica. El problema no es la herramienta, sino la falta de formación especializada en su uso. ARIA es una especificación técnica densa y rigurosa que no perdona errores. A diferencia del HTML, donde un error de sintaxis a menudo es ignorado por el navegador, permitiendo que el contenido se visualice, un error en ARIA puede alterar drásticamente la forma en que un usuario percibe la página.

Un ejemplo claro de esto es el mal uso del atributo `role="application"`. Cuando se aplica incorrectamente a un elemento contenedor amplio, como el cuerpo del documento, puede desactivar las teclas de navegación estándar del lector de pantalla, dejando al usuario ciego en un limbo funcional donde sus comandos habituales no responden. Este tipo de error catastrófico es lo que alimenta el miedo a usar ARIA. Sin embargo, la solución no es prohibir el atributo, sino educar sobre su propósito específico: indicar regiones de la interfaz que requieren un manejo de teclado personalizado, similar al de una aplicación de escritorio. De igual manera, el uso indiscriminado de regiones en vivo, mediante `aria-live` o `role="alert"`, puede transformar una experiencia de navegación tranquila en un caos auditivo. Configurar una región como `aria-live="assertive"` interrumpe cualquier cosa que el lector de pantalla esté verbalizando en ese momento. Si se usa para notificaciones triviales, se genera frustración y fatiga cognitiva en el usuario. Por el contrario, si no se utiliza ARIA para notificar cambios dinámicos en la pantalla —algo intrínseco en las WebApps—, la persona que necesita accesibilidad nunca se enterará de que el contenido ha cambiado tras pulsar un botón, lo que es igualmente bloqueante.

La solución

El discurso profesional de la accesibilidad debe madurar. Debemos abandonar la simplificación excesiva que clasifica a ARIA como el enemigo y comenzar a tratarla como una herramienta de alta precisión necesaria para la ingeniería web avanzada. La responsabilidad de los consultores y expertos en accesibilidad es fomentar una formación técnica profunda. Los desarrolladores deben entender que ARIA no es un parche para un mal HTML, sino una capa adicional de semántica necesaria cuando el HTML llega a su límite. Es necesario enseñar no solo la sintaxis, sino la ontología de los estados y propiedades, el ciclo de vida de los componentes accesibles y las implicaciones de cada atributo en el árbol de accesibilidad.

El desarrollo de componentes accesibles con ARIA requiere un conocimiento que va más allá de la mera lectura de la especificación; requiere comprender cómo los diferentes navegadores mapean estos atributos a las API de accesibilidad del sistema operativo —como UI Automation en Windows o AXAPI en macOS— y cómo los distintos lectores de pantalla interpretan esa información. Es una disciplina que combina la ingeniería de software con la ergonomía y la interacción persona-ordenador. Al demonizar ARIA, desalentamos a los desarrolladores de adquirir este conocimiento crucial, manteniendo el ciclo de ignorancia y mala implementación.

El futuro de la web es innegablemente rico, interactivo y orientado a aplicaciones. Las tecnologías de asistencia y los estándares web están convergiendo para hacer posible que estas experiencias sean inclusivas. ARIA evoluciona continuamente para dar soporte a nuevas capacidades de la plataforma web, como los modelos de objetos accesibles y las nuevas primitivas de control. Los profesionales de la accesibilidad deben liderar este cambio, promoviendo el estudio riguroso de ARIA como una competencia esencial de diseñadores y desarrolladores. Debemos enseñar a detectar los nuevos problemas que surgen en las interfaces complejas —como las trampas de foco, la falta de retroalimentación en cambios de estado asíncronos o la inconsistencia en los nombres accesibles— y demostrar cómo ARIA, aplicada con conocimiento y precisión, es la llave maestra para solucionarlos.

El camino hacia una web accesible pasa inevitablemente por aprender a usar ARIA correctamente, reconociéndola como el aliado indispensable que garantiza que la innovación tecnológica no deje a nadie atrás.

El camino hacia una web accesible pasa inevitablemente por aprender a usar ARIA correctamente, reconociéndola como el aliado indispensable que garantiza que la innovación tecnológica no deje a nadie atrás.

Imagen 1: guerrero con escudo con la A de Aria creada con IA.