WAI-ARIA

WAI-ARIA **1**

Roles 2

Propiedades e estados 3

¿Onde está soportado WAI-ARIA? 4

¿Cando se debe usar WAI-ARIA? **4**

Implementacións prácticas de WAI-ARIA **5**

Puntos de referencia 5

Actualización de contido dinámico 6

Mellora a accesibilidade do teclado 6

Accesibilidade de controis non semánticos 7

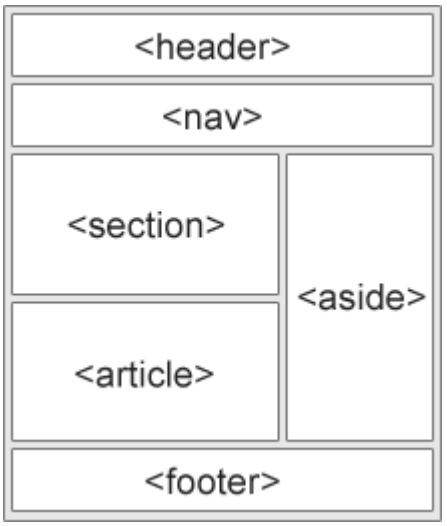
Referencias **8**

WAI-ARIA

WAI-ARIA (*Web Accesibility Initiative - Accessible Rich Internet applications - Aplicacións de Internet Ricas en Accesibilidade*) é unha especificación do W3C. É unha tecnoloxía que permite engadir información semántica adicional ao código HTML para que os navegadores e as tecnoloxías de apoio poidan recoñecer compoñentes e usalos.

A medida que as aplicacións web comenzaron a ser máis complexas e dinámicas, un novo conxunto de características e problemas de accesibilidade empezaron a aparecer.

HTML5 introduciu elementos semánticos para definir características comúns dunha páxina (<nav>, <footer>, <section>, etc.). Antes de que estes elementos estivesen dispoñibles, creábanse páxinas web utilizando, por exemplo, , pero estes eran problemáticos, xa que non era fácil encontrar automaticamente unha característica específica da páxina como o menú de navegación principal.



Os elementos semánticos de HTML5 describen o seu significado ao navegador e á persoa que desenvolve o código HTML. Tamén resultan útiles ás tecnoloxías de apoio para interpretar o significado do elemento.

WAI-ARIA permite incluír información semántica sobre a estrutura da páxina de forma que os produtos de apoio poidan anunciar, acceder e saltar polos bloques relevantes da páxina: cabeceira, zona de navegación, contido principal, pé de páxina, etc.

Outro exemplo onde WAI-ARIA é de axuda é na utilización de controis complexos: selector de data, slider, etc. HTML5 proporciona compoñentes especiais para estes controis.

- <input type="date">
- <input type="range">

Estes controis non son soportados en todos os navegadores, polo que ás veces se programan utilizando bibliotecas de JavaScript que xeran controis con unha serie de , código CSS e controlados por JavaScript. O problema aquí é que visualmente funcionan, pero os lectores de pantalla non son capaces de interpretalos, xa que non teñen contido semántico que os describa. WAI-ARIA permite definir que rol ou función ten un elemento e indicar o seu estado e propiedades. Por exemplo, pode indicarse que un DIV é en realidade un menú despregable, se está pregado ou non e cambiar o seu estado cando as persoas usuarias interactúan con el.

WAI-ARIA permite incluír información semántica sobre:

- a estrutura da páxina.
- os compoñentes (por exemplo, se un campo dun formulario é obrigatorio), a interface, o seu comportamento e a relación entre os mesmos.

Esta información semántica pode ser transmitida ás persoas que utilizan produtos de apoio, como un lector de pantalla, e facilitar así a comprensión da páxina e a interacción coa mesma.

WAI-ARIA está pensada para facer máis accesible o contido dinámico e os controis creados con Ajax, HTML, Javascript e tecnoloxías relacionadas.

Funcionalmente, a información que engade ARIA é como un CSS para as tecnoloxías de apoio. É dicir, para un lector de pantalla, ARIA controla a renderización da experiencia non visual.

WAI-ARIA permite describir case calquera compoñente dunha forma que as tecnoloxías de apoio poden interpretar con fiabilidade.

Para incluír esta información semántica sobre a interface e o seu comportamento, WAI ARIA proporciona unha ontoloxía de roles, estados e propiedades:

Roles

Un rol define un elemento e serve para indicar de que tipo é e cal é a súa función.

Especificar o rol dun elemento dá información á tecnoloxía de apoio para saber como tratar ese elemento.

Os roles están clasificados en varias categorías, que poden englobarse en dous grandes grupos:

- roles que definen elementos da interface (*widget roles*) como botóns, barras de progreso, sliders, searchbox, etc.
- roles que definen a estrutura da páxina como seccións, elementos de navegación, etc. As persoas usuarias dun lector de pantalla poderán saber cal é estrutura da páxina usando estes roles, e poderán saltar ao bloque de contido no que estean interesadas.

Incluír un rol a un elemento é tan sinxelo como engadir o atributo “role” ao elemento e indicar un rol dos definidos na especificación.

Exemplo:

**<ul role=”tree”>...
**

O rol “tree” define unha lista que se comporta como unha árbore despregable.

Moitos roles duplican o valor semántico dos elementos HTML5, como **role=”navigation”** (<nav>) ou **role=”complementary”** (<aside>), pero hai outros que describen diferentes estruturas da páxina, como **role=”banner”**, **role=”tabgroup”**, **role=”tab”**, etc.

Actualmente os lectores de pantalla poden anunciar, acceder e saltar polos bloques da páxina: cabeceira, zona de navegación, contido principal, buscador ou pé de páxina. Para que isto sexa posible, hai que marcar esas zonas co rol adecuado.

Propiedades e estados

As propiedades e estados son atributos. En realidade teñen características similares, ambos proporcionan información específica sobre un obxecto. WAI-ARIA diferénciaos conceptualmente indicando que as propiedades soen cambiar menos (aínda que non sempre) que os estados, que cambian con frecuencia debido á interacción das persoas usuarias.

As propiedades son atributos que son esenciais á natureza dun obxecto dado, ou que representan valores asociados co obxecto. Normalmente as propiedades dun elemento non cambian de valor, ao contrario dos estados que si o fan.

As propiedades úsanse para dar significado extra. Por exemplo, **aria-required=”true”** especifica que un campo input dun formulario debe ser completado para ser válido, mentres que **aria-labelledby=”label”** permite referenciar un elemento identificado con un ID, incluso múltiples elementos, o que non sería posible usando **<label for=”input”>**.

Exemplo:

```
<div role="img" aria-labelledby="caption">
  <img src="example.png" role="presentation" alt="">
  <p id="caption">A visible text caption labeling the image.</p>
</div>
```

Un **estado** é unha propiedade dinámica que expresa unha característica dun obxecto que pode cambiar en resposta á interacción das persoas usuarias ou automaticamente.

O estado é unha propiedade especial que define a condición actual dun elemento, como por exemplo **aria-disabled=”true”**, que especifica a un lector de pantalla que un campo de entrada está deshabilitado. Os estados difiren das propiedades en que as propiedades non cambian ao longo do ciclo de vida da aplicación, mentres que os estados poden cambiar, xeralmente usando JavaScript.

Exemplo:

```
<li role="menuitemcheckbox" aria-checked="true">Sort by Last Modified</li>
```

Os cambios que se produzan tanto nos estados como nas propiedades dun elemento son notificados ás tecnoloxías de apoio que poden alertar ás persoas usuarias de que se produciu un cambio.

Lista de estados e propiedades coa súa definición

Un punto importante sobre os atributos WAI-ARIA é que estes non afectan para nada á páxina web, excepto a información exposta polas APIs de accesibilidade do navegador (de onde os lectores de pantalla collen a súa información). WAI-ARIA non afecta á estrutura da páxina, nin ao DOM, etc.

¿Onde está soportado WAI-ARIA?

Esta é unha pregunta difícil de responder porque:

- WAI aria ten moitas características na súa especificación.
- Hai moitas combinacións posibles de sistema operativo, navegador e lector de pantalla a considerar.

O soporte do navegador é bastante bo. Actualmente, consultada a web caniuse.com, esta declara un soporte global do navegador para WAI-ARIA do 96,88%.

O soporte do lector de pantalla para as funcións de ARIA non está neste nivel. Pode consultarse o artigo de Compatibilidade do lector de pantalla con WAI-ARIA.

¿Cando se debe usar WAI-ARIA?

Anteriormente describíronse problemas que fixeron que WAI-ARIA aparecese, pero realmente, hai catro áreas principais nas que WAI-ARIA é útil:

- **Puntos de referencia**: os valores do atributo **role** poden facer de puntos de referencia de áreas funcionais: **search, tab, menu, listbox,** etc.
- **Actualizacións de contido dinámica:** os lectores de pantalla tenden a ter dificultades para informar de contido que se actualiza constantemente. Con ARIA pódese usar **aria-live** para informar aos lectores de pantalla cando un área de contido é actualizada.
- **Mellorar a accesibilidade do teclado**: hai elementos propios de HTML que teñen accesibilidade de teclado de forma nativa; cando se utilizan outros elementos xunto con JavaScript para simular interaccións similares, a accesibilidade do teclado e os lectores de pantalla sofren. Cando isto é inevitable, WAI-ARIA proporciona métodos para permitir aos elementos recibir o foco (usando **tabindex**).
- **Accesibilidade de controis non semánticos**: cando se crea un elemento complexo da interface de usuario usando aniñados, CSS e JavaScript, a accesibilidade vese comprometida. É difícil para un lector de pantalla saber o funcionamento do elemento se non hai semántica ou outras pistas. Nesta situación, ARIA pode axudar engadindo o significado que falta cunha combinación de roles como **button, listbox** ou **menu**, e propiedades como **aria-required** ou **aria-posinset** (define un número ou

posición para un elemento dentro dun conxunto) para proporcionar máis pistas sobre a funcionalidade.

Recordar: só usar WAI-ARIA cando sexa necesario. Idealmente, debe usarse a característica nativa de HTML para proporcionar a semántica requirida polos lectores de pantalla. Cando isto non sexa posible é cando WAI-ARIA pode ser unha valiosa ferramenta para mellorar a accesibilidade.

Implementacións prácticas de WAI-ARIA

Neste apartado veranse as catro áreas nas que WAI-ARIA é útil, con exemplos prácticos.

Debería utilizarse un lector de pantalla para comprobar os exemplos.

Puntos de referencia

O atributo role permite engadir valor semántico extra aos elementos web.

O seguinte exemplo mostra como engadir información adicional aos lectores de pantalla:

- Exemplo sen WAI-ARIA.
- Exemplo con WAI-ARIA.

```
<header>
  <h1>...</h1>
  <nav role="navigation">
    <ul>...</ul>
    <form role="search">
      <!-- search form  →
          <input type="search" name="q" placeholder="Search query" aria-label="Search
through site content">
    </form>
  </nav>
</header>

<main>
  <article role="article">...</article>
  <aside role="complementary">...</aside>
</main>

<footer>...</footer>
```

No exemplo anterior engadíronse atributos de tipo rol á estrutura HTML. Ademais, o elemento <input> ten un atributo **aria-label** que proporciona unha etiqueta para ser lida polo lector de pantalla.

Actualización de contido dinámico

Por defecto, cando se actualiza de forma dinámica un contido nunha páxina, o lector de pantalla non o detecta, polo que as persoas usuarias non saben o que está pasando (imaxinar por exemplo un chat).

Utilizando a propiedade **aria-live** proporciónase un mecanismo para proporcionar alertas ante o cambio de contido. Esta propiedade pode ter os seguintes valores:

- **off:** valor por defecto. As actualizacións non son anunciadas.
- **polite:** as actualizacións son anunciadas só se a persoa usuaria está ociosa.
- **assertive:** as actualizacións son anunciadas tan pronto como sexa posible.

Exemplo en funcionamento:

```
<section aria-live="assertive" aria-atomic="true">
...
```

Co exemplo anterior, o lector de pantalla lerá o contido cando se actualice. Por defecto só se le o contido que se actualiza, aínda que sería interesante ler tamén a cabeceira para que a persoa saiba que se está lendo. Para facer isto engadiuse a propiedade **aria-atomic=”true”** que lle indica ao lector que lea o elemento por completo como unha unidade atómica, non só o contido actualizado.

Mellora a accesibilidade do teclado

HTML proporciona accesibilidade nativa do teclado en botóns, controis de formularios, ligazóns. Xeralmente pode usarse o tabulador para moverse entre controis, a tecla Enter para seleccionar ou activar controis.

Sen embargo, algunhas veces hai código que utiliza elementos non semánticos como botóns (ou outros controis). Recoméndase non utilizar este tipo de código, mais ás veces é herdado de aplicacións antigas.

WAI-ARIA permite que elementos que non poidan ter o foco, poidan recibilo utilizando o atributo **tabindex.**

- tabindex=”0” para que se poida acceder a dito elemento por teclado mediante tabulador, na orde secuencial de navegación do teclado. A orde de tabulación depende do contido do documento.
- tabindex=”-1” significa que o elemento non pode coller o foco a través do teclado pero si a través de JavaScript ou pulsando co rato. Úsase para crear widgets accesibles con JavaScript.
- Un valor positivo significa que o elemento debe coller o foco na orde secuencial de navegación. A orde está definida polo valor do número. É dicir, tabindex="4" colle o foco antes de tabindex="5" e tabindex="0", pero despois de tabindex="3". Se

múltiples elementos comparten o mesmo valor positivo, a súa orde segue a súa posición no documento. O valor máximo de tabindex é 32767. Se non se especifica, o valor por defecto é 0.

Exemplo onde se fai que os botóns (creados con div) teñan o foco.

Accesibilidade de controis non semánticos

Cando se crea un elemento complexo da interface de usuario usando aniñados, CSS e JavaScript, a accesibilidade vese comprometida. É difícil para un lector de pantalla saber o funcionamento do elemento se non hai semántica ou outras pistas. Nestas situacións, ARIA pode axudar engadindo o significado que falta.

- Validación de formularios e mensaxes de alerta. Neste exemplo móstrase como indicar as mensaxes de erro resultantes da validación dun formulario.

```
<div class="errors" role="alert" aria-relevant="all">
  <ul>
  </ul>
</div>
```

role=”alert” fai que os cambios no elemento sexan lidos por un lector de pantalla. Ao mesmo tempo identifícase con unha mensaxe de alerta nun formato máis accesible (as mensaxes **alert()** teñen problemas de acccesibilidade).

aria-relevant=”all” fai que o lector en pantalla lea o contido do elemento cando este cambie.

¿Como indicar que un campo dun formulario é obrigatorio?

Normalmente faise con un <label> e un asterisco, pero isto non é fácil de entender para os lectores de pantalla.

```
<input type="text" name="name" id="name" aria-required="true">
```

aria-required=”true” da pistas ao lector de pantalla para indicar que campos deben ser cubertos.

Nalgúns campos tamén é necesario indicar o rango de valores posibles:

```
<input type="number" name="age" id="age" placeholder="Enter 1 to 150"
aria-required="true">
```

O atributo **placeholder** pode ter unha mensaxe que é lida polos lectores de pantalla.

A versión actualizada do exemplo pode encontrarse na seguinte ligazón.

- Describir botóns non semánticos como botóns.

  Cando se crea un botón falso (usando por exemplo), pode engadirse accesibilidade de teclado usando **tabindex**. Aínda así, os lectores de pantalla non os identifican como botóns. Isto pode resolverse engadindo **role** ao elemento.

```
<div data-message="This is from the first button" tabindex="0"
role="button">Click me!</div>
```

  Deque university code library contén exemplos de como facer accesibles diversos controis.

Referencias

Para a elaboración deste material utilizáronse, entre outros, os recursos que se enumeran a continuación:

- https://developer.mozilla.org/en-US/docs/Learn/Accessibility/WAI-ARIA_basics
- https://www.w3.org/TR/wai-aria-practices/
- https://olgacarreras.blogspot.com/2007/09/wai-aria-introduccion-referencias.html