Actividad 5.1

ARIA

Pon en práctica los aspectos más relevantes de ARIA como apoyo al desarrollo de contenido Web Accesible.
Utiliza la web https://liveweave.com/ para poner a pruebas los distintos ejercicios. Descarga la web cuando debas usar alguna herramienta de validación o comprobación de accesibilidad.

Recuerda
Accessible Rich Internet Applications (**ARIA**) define cómo realizar contenido Web y aplicaciones Web (especialmente las desarrolladas con Ajax y JavaScript) más accesibles a personas con discapacidades. Por ejemplo, ARIA posibilita puntos de navegación accesibles, widgets JavaScript, sugerencias en formularios y mensajes de error, actualizaciones en directo, y más.

Reglas de uso

Antes de usar ARIA recuerda las reglas de uso:

1. Si puede usar un elemento **HTML** nativo [HTML51] o atributo con la semántica y el comportamiento que necesita ya incorporado, en lugar de reutilizar un elemento y agregar un rol, estado o propiedad **ARIA** para hacerlo accesible, hágalo.
   ¿En qué circunstancias puede que esto no sea posible?
   a. Si la función está disponible en HTML [HTML51] pero no está implementada o está implementada, pero el soporte de accesibilidad no lo está.
   b. Si las restricciones de diseño visual descartan el uso de un elemento nativo en particular, porque el elemento no se puede diseñar como se requiere.
   c. Si la función no está disponible actualmente en HTML.
2. No cambie la semántica nativa de las etiquetas, por ejemplo, se quiere crear un encabezado que sea una pestaña:

```
<h2 role=tab>heading tab</h2>
```

```
<div role=tab><h2>heading tab</h2></div>
```

3. Todos los controles ARIA interactivos deben poder utilizarse con el teclado. Si crea un widget en el que un usuario puede hacer clic, tocar, arrastrar, soltar, deslizar o desplazarse, el usuario también debe poder navegar hasta el widget y realizar una acción equivalente utilizando el teclado.
   Todos los widgets interactivos deben estar programados para responder a las pulsaciones de teclas estándar o combinaciones de pulsaciones de teclas cuando corresponda.
   Por ejemplo, si se utiliza role=button el elemento debe ser capaz de recibir el foco y el usuario debe ser capaz de activar la acción asociada con el elemento utilizando tanto el **enter** (en Windows) o **return** (MacOS) y la tecla espaciadora.

4. No utilice role="presentation" o aria-hidden="true" sobre un elemento enfocable.

```
<button role=presentation>Presionar</button>
<button aria-hidden="true">Presionar</button>
```

```
<div aria-hidden="true">
    <button>press me</button>
</div>
```

```
button {opacity:0}
<button tabindex="-1" aria-hidden="true">Presionar</button>
```

5. Todos los elementos interactivos deben tener un nombre accesible. Un elemento interactivo solo tiene un nombre accesible cuando su propiedad de nombre accesible de API de accesibilidad (o equivalente) tiene un valor. Por ejemplo, input type=text en el ejemplo de código a continuación tiene una etiqueta visible *'nombre de usuario'*, pero no un nombre accesible:

```
<!-- Nota: usa for/id o label  -->
<input type="text" aria-label="Nombre de usuario">
or
<span id="p1">Nombre de usuario</span> <input type="text" aria-labelledby="p1">
```

Reto 1: aria-label, aria-labelledly y aria-describedby

Estos atributos nos permiten dar un nombre accesible, etiquetar o dar una descripción a los elementos HTML que lo necesiten.

a) Crea un enlace <a> y descríbelo con arial-label.
b) Utiliza **total11y** para comprobar que la etiqueta se describe correctamente.
c) Crea tres enlaces <a> para descargar los documentos guia.pdf, guia.docx y guia.pptx. Utiliza aria-labelledby para describir los enlaces según el tipo de documento y utilizando una descripción general que esté dentro de un elemento <p>.

   Ejemplo: PDF | Word | Powerpoint
d) Utiliza **total11y** para comprobar que la etiqueta se describe correctamente.
e) Copia el código siguiente en una página HTML y comprueba el funcionamiento de ARIA.

```
<input aria-labelledby="labelShutdown shutdownTime shutdownUnit" type="checkbox" />
<span id="labelShutdown">Shut down computer after</span>
<input aria-labelledby="labelShutdown shutdownTime shutdownUnit" id="shutdownTime" type="text" value="10" />
<span id="shutdownUnit"> minutes</span>
```

Describe cómo se genera el texto alternativo en este caso. Compruébalo con total11y cambiando el valor del input y verificando el texto alternativo del *checkbox*.

f) Comprueba el funcionamiento del siguiente control:

```
<div aria-describedby="test">Texto de ejemplo</div>
<div id="test" role="tooltip" >Esto es un tooltip de ejemplo</div>
```

Reto 2: aplica estilos usando elementos de ARIA

Aplica el estilo siguiente:

```
div[role="navigation"] { color: green; background-color: inherit; }
```

A la sección a continuación:

```
<div id="navigation" role="navigation">
  <ul>
    <li id="active"><a id="current" href="home">Inicio</a></li>
    <li><a href="blog">Blog</a></li>
    <li><a href="contacto">Contacto</a></li>
  </ul>
</div>
```

¿Cuál sería el elemento para ARIA en este caso?

Reto 3: ARIA live regions

Suponga que tiene un elemento TEXT INPUT con un límite máximo de 20 caracteres. Es necesario asociar a este elemento un SPAN que sirva de contenedor para mostrar al usuario el número restante de caracteres que le quedan por escribir. Como sabrá, con JavaScript podrá ir restando y actualizando el contenido de este elemento, que hace de contador.

Este sería el código HTML para este supuesto:

```
<label for="user-name">
    Nombre (obligatorio)
    <input name="user-name" id="user-name" type="text" size="20" maxlength="20" />
    <span id="count"></span>
</label>
```

Un usuario con discapacidad visual no tendrá acceso a la información de ese contador puesto que las tecnologías asistivas no reflejarán la actualización del elemento. Explica la solución ARIA a continuación.

```
<label for="user-name">
    Nombre (obligatorio)
    <input name="user-name" id="user-name" type="text" size="20" maxlength="20" aria-required="true" />
    <span id="count" aria-live="polite"></span>
</label>
```

Parte I

Con JavaScript, es posible cambiar dinámicamente partes de una página sin tener que recargar toda la página; por ejemplo, para actualizar una lista de resultados de búsqueda sobre la marcha o para mostrar una alerta o notificación discreta que no requiere la interacción del usuario.

Si bien estos cambios suelen ser visualmente evidentes para los usuarios que pueden ver la página, es posible que no sean obvios para los usuarios de tecnologías de asistencia. Las regiones en vivo de ARIA llenan este vacío y brindan una manera de exponer de manera programática los cambios de contenido dinámico de una manera que puede ser anunciada por tecnologías de asistencia.

> Las tecnologías de asistencia anunciarán cambios dinámicos en el contenido de una región en vivo. Incluir un atributo aria-live o un rol de región en vivo especializado (como **role="alert"**) en el elemento en el que desea anunciar cambios funciona siempre que agregue el atributo antes de que ocurran los cambios, ya sea en el marcado original o dinámicamente usando JavaScript.

aria-live: aria-live=POLITENESS_SETTING se usa para establecer la prioridad con la que el lector de pantalla debe tratar las actualizaciones de las regiones en vivo; las configuraciones posibles son: **off**, **polite** o **assertive**. La configuración predeterminada es **off**. Este atributo es, con mucho, el más importante.

- Normalmente, solo se usa **aria-live="polite"**. Cualquier región que reciba actualizaciones que sean importantes para el usuario, pero que no sean tan rápidas como para ser molestas, debería recibir este atributo. El lector de pantalla hablará los cambios siempre que el usuario esté inactivo.
- **aria-live="assertive"** solo debe usarse para notificaciones urgentes / críticas que requieren absolutamente la atención inmediata del usuario. Generalmente, un cambio a una región en vivo asertiva interrumpirá cualquier anuncio que esté haciendo un lector de pantalla. Como tal, puede ser extremadamente molesto y perturbador y solo debe usarse con moderación.
- Como **aria-live="off"** es el valor predeterminado asumido para los elementos, no debería ser necesario establecer esto explícitamente, a menos que esté intentando suprimir el anuncio de elementos que tienen un rol de región en vivo implícito (como **role="alert "**).

A medida que el usuario selecciona un nuevo planeta, el lector de pantalla anunciará la información de la región activa. Debido a que la región en vivo tiene **aria-live="polite"**, el lector de pantalla esperará hasta que el usuario haga una pausa antes de anunciar la actualización.
Realice las siguientes pruebas sobre el ejercicio (ejercicio_051):

- Utiliza **total11y** para comprobar la existencia del contenido descriptivo.
- Activa el narrador de Windows para comprobar el cambio de texto descriptivo de forma dinámica. Otra alternativa es usar el ChromeVox.

Parte II

- **aria-atomic: aria-atomic=BOOLEAN** se usa para establecer si el lector de pantalla debe presentar siempre la región activa como un todo, incluso si solo cambia una parte de la región. Los posibles ajustes son: false o true. La configuración por defecto es false.
- **aria-relevante: el aria-relevante = [LIST_OF_CHANGES]** se utiliza para establecer qué tipos de cambios son relevantes para una región activa. Las configuraciones posibles son una o más de: additions, removals, text, all. La configuración predeterminada es: additions text.

Comprueba el funcionamiento de la página y alterna con el uso de **aria-atomic** en el (ejercicio_052)

```
<h2>Reloj JS</h2>
<div id="clock" role="timer" aria-live="polite">
<!--div id="clock" role="timer" aria-live="polite" aria-atomic="true"-->
    <span id="clock-hours"></span>
    <span id="clock-mins"></span>
</div>
```

Parte III

Comprueba el funcionamiento de la página del (ejercicio_053).

Reto 4: ARIA y JQuery

Copia el código siguiente dentro del *<body>* de la plantilla de https://liveweave.com/

```
<div id="head">
    <h1 id="logo">
      <img src="http://clipart-library.com/images/rTLxGokzc.jpg" alt="Título" />
      <span id="tagline">Descripción</span>
    </h1>
    <ul id="nav">
      <li><a href="/">home</a></li>
      <li><a href="/blog">blog</a></li>
      <li id="active"><a id="current" href="/contact">contact</a></li>
    </ul>
    <form id="searchform" method="get">
      <fieldset>
        <legend>Search</legend>
        <label for="s">
          <input type="text" name="s" id="s" value="Search" />
        </label>
        <label for="srhsub">
          <input type="image" name="srhsub" id="srhsub" src="http://clipart-
library.com/images/8iE6KxzyT.jpg" value="seek" alt="seek" />
        </label>
      </fieldset>
    </form>
  </div>

  <div id="main">
    <h2>Cuerpo</h2>
    <p>Cotenido</p>
  </div>

  <div id="footer">
    <p id="copy">Accesibilidad 2021</p>
  </div>
```

Añade el siguiente fragmento javascript:

```
$(document).ready(function() {
          $('#logo').attr('role', 'banner');
          $('#nav').attr('role', 'navigation');
          $('#searchform').attr('role', 'search');
          $('#main').attr('role', 'content');
          $('#footer').attr('role', 'contentinfo');
          $('.required').attr('aria-required', 'true');
});
```

Con la herramienta total11Ay comprueba qué cambia en el formulario y cómo contribuye la accesibilidad.

Reto Extra: recursos de la web

Con ayuda de ChromeVox, comprueba los recursos dinámicos disponibles en la siguiente web [usableyaccesible - ejemplo aria live regiosn](#).

==========

Algunas soluciones mezcladas

```
<a href="#" aria-label="Cerrar">X</a>

<p id="titulo-informe">Descargar el informe de accesibilidad en:</p>

<a aria-labelledby="titulo-informe pdf" href="pdf.pdf" id="pdf">PDF</a> |

<a aria-labelledby="titulo-informe docx" href="word.docx" id="docx">Word</a> |

<a aria-labelledby="titulo-informe pptx" href="ppt.pptx" id="pptx">Powerpoint</a>
```

Consideraciones

- aria-labelledby y aria-describedby cuentan con un sólido soporte para elementos de contenido interactivo como enlaces y controles de formulario, incluidos los muchos input tipos.
- Para la mayoría de tecnología de asistencia que está bien para el uso aria-label o aria-labelledby en el <nav>, y <main> elementos, pero no en <footer>, <section>, <article>, o <header>.
- Hay un apoyo mixto para aria-label o aria-labelledby sobre <aside>.
- Talkback en Android anula el contenido de todos los puntos de referencia con aria-label o aria-labelledby.
- Cuidado con el uso de aria-label o aria-labelledby en div con role=navigation, role=search, role=main, JAWS no es compatible con ellos en role=banner, role=complementary, role=contentinfo. NVDA y VoiceOver funcionan bien.
- aria-label, aria-labelledby y aria-describedby funcionan bien en table, th y td con algunas excepciones para NVDA, VoiceOver en iOS.
- No uses aria-label ni aria-labelledby en ningún elemento de encabezado porque los anula en NVDA, VoiceOver y Talkback. JAWS los ignora.
- No utilice aria-label o aria-labelledby en cualquier otro contenido que no sea interactivo, como p, legend, li, o ul, ya que se ignora.
- No use aria-label o aria-labelledby en un span o a div ***menos que se*** le ***haya*** dado un role. Cuando aria-label o aria-labelledby están en roles interactivos (como un link o button) o un img rol, anulan el contenido del div o span.
- aria-describedby en un span o div será ignorado por NVDA y VoiceOver a menos que se le proporcione una imagen interactiva, un role o un punto de referencia role. JAWS y Talkback son compatibles.
- aria-describedby NVDA y VoiceOver lo ignorarán en cualquier otro contenido estático. JAWS y Talkback lo interpretan bien.

Todo lo anterior también funciona igual en elementos iframe . Ambos aria-label y aria-labelledby tienen el mismo comportamiento con los lectores de pantalla y la API de accesibilidad, pero aria-label debe reservarse para cuando no haya texto visible en la página para hacer referencia o cuando sea demasiado difícil realizar un seguimiento de los valores de identificación.