AUTOR

Ingeniero informático, músico compositor, filántropo, escritor y profesor.

A la edad de 14 años descubrí mi vocNotsación casi por accidente y, un año más tarde, empecé a programar de forma autodidacta. Pocos años después, en 1992, realicé un software de aprendizaje de cardiología para los estudiantes de la Universidad Complutense de Medicina de Madrid mientras continuaba formándome como profesional. Posteriormente estuve dando clases a jóvenes en la academia Santillana, trabajando como Administrador de Sistemas y como Técnico de reparación de ordenadores hasta que, en 1996, empecé la Ingeniería Técnica de Sistemas Informáticos.

Paralelamente, empecé a realizar proyectos de I+D sobre nuevas tecnologías, redes sociales, servicios, e-commerce, seguridad, SEO y movilidad con fines no lucrativos. Después de muchos altibajos que marcaron mi vida personal y profesional en 2008 empecé a trabajar para Hewlett Packard como Full Stack Developer en Soluciones Integrales de Administración IT, administración de sistemas y desarrollo de aplicaciones web. Desde 2011 hasta 2014 estuve trabajando en varias empresas realizando diseño, desarrollo interfaces de programación de aplicaciones, integración con otras plataformas y servicios, SEO Orgánico, analítica Web y análisis funcional hasta que aterricé en Sopra-Steria dónde poco a poco fui entrando en la Experiencia de Usuario, Usabilidad y Accesibilidad Web hasta que, actualmente, soy el Responsable de UX del Sector Público y Retail.

1

DISEÑO CENTRADO EN LA ACCESIBILIDAD

"Un sistema usable puede que no sea accesible pero un sistema accesible, seguro que es usable"

Pablo E. Fernández

1.1 QUÉ ES LA ACCESIBILIDAD WEB

La accesibilidad es la cualidad de accesible, un adjetivo que se refiere a aquello que es de fácil acceso, trato o comprensión. El concepto se utiliza para nombrar al grado en el que todas las personas, más allá de sus capacidades físicas o técnicas, pueden utilizar un cierto objeto o acceder a un servicio.

Existen diversas ayudas técnicas para promover la accesibilidad y equiparar las posibilidades de todas las personas. Esto supone que un lugar que presenta buenas condiciones de accesibilidad puede recibir a toda clase de gente sin que exista un perjuicio o una dificultad para nadie.

Una de estas ayudas técnicas más comunes es lo que se denomina tecnología asistiva. **Una tecnología asistiva** (TA) es una herramienta utilizada para permitir que personas o usuarios con discapacidad puedan beneficiarse de las mismas ventajas que sus pares sin discapacidad.

Cuando se habla accesibilidad Web, en realidad, se hace referencia a una serie de normas de diseño que van a permitir a todo tipo de usuarios (con o sin discapacidad) percibir, entender, navegar e interactuar con una interfaz o sistema.

Un grupo de estas normas se conocen como **Pautas de Accesibilidad para Agentes de Usuario** (UAAG) y muestran cómo hacer que las herramientas formadas

por navegadores, reproductores multimedia y tecnologías asistivas, entre otras, sean accesibles para personas con discapacidad.

Otro grupo de normas son las denominadas **Pautas de Accesibilidad para Herramientas de Autor** (ATAG) y tienen como objetivo definir la forma en la que las herramientas ayudan a los desarrolladores o diseñadores a producir un contenido que cumpla todas las Pautas de Accesibilidad al Contenido en la Web (WCAG).

Las ATAG están pensadas principalmente para desarrolladores entre las que se incluyen:

- Editores de HTML y XML de WYSIWYG (What You See Is What You Get).
- Procesadores de texto o paquetes de publicación.
- Herramientas de conversión que transforman formatos de publicación a HTML.
- Edición y producción de vídeo, paquetes de autor de SMIL.
- Gestores de contenido (CMS), herramientas de conversión instantánea o de publicación de sitios Web.
- Herramientas de diseño (SASS, SVG o gráficos vectoriales, minificadores, …).

1.1.1 Tipos de discapacidad

Actualmente, muchos de los sistemas, por no decir la mayoría, son inaccesibles (en mayor o menor media) lo que dificulta o imposibilita la utilización Internet para muchos usuarios con discapacidad.

La accesibilidad Web engloba los tipos de discapacidades en cuatro grandes grupos:

- La **discapacidad visual** es una anomalía parcial o total del sentido de la vista y que puede referirse desde a una pérdida de visión hasta a una sensibilidad especial a la fotografía o a la luz.
- La **discapacidad física** es un tipo de anomalía que imposibilita o dificulta, a quien la padece, el control de las funciones motoras o de su cuerpo.
- La **discapacidad auditiva** es una anomalía parcial o total del sentido del oído y que puede referirse desde a una pérdida de audición parcial, lo que se denomina hipoacusia, hasta a una pérdida total, lo que se conoce como cofosis.
- La **discapacidad intelectual** es una anomalía que imposibilita o dificulta realizar funciones de tipo mental como es el habla, el cuidado personal o la

integración social y no tiene por qué estar asociada a ninguna enfermedad o trastorno ya que, mucha de la población mundial, tiene algún tipo de discapacidad intelectual. También se la suele denominar **discapacidad cognitiva** si va referida al desarrollo intelectual y/o la adaptación social de algunas personas.

Viendo la cantidad de discapacidades que existen y la cantidad de usuarios que poseen una o varias de ellas, se hace imperioso la necesidad de suministrar accesibilidad a las interfaces o sistema. No sólo porque aumente su usabilidad, ni porque pueda tener mejor indexación con los motores de búsqueda, sino porque lo importante son los usuarios.

1.2 TECNOLOGÍAS DÓNDE LA ACCESIBILIDAD WEB ES APLICABLE

HTML (HyperText Markup Language) y XHTML (eXtensible HTML)

Los lenguajes de marcado HTML y XHTML pueden ser buenos recursos a la hora de hacer una web accesible.

Mientras que HTML está basado en la tecnología denominada Standard Generalized Markup Language (SGML; ISO 8879: 1986), XHTML está basado en Extensible Markup Language, también conocido como XML. La principal diferencia es que XHTML es mucho más estricto y, por ello, algunos métodos pueden ser mucho más difíciles de conseguir, sin contar que, XHTML, no es semántico.

Realizar una web semántica no implica más tiempo de desarrollo, ni más coste que una web no semántica. De hecho, cuando se aplican estructuras semánticas, el desarrollo se vuelve más fácil con el tiempo, se mejora el Posicionamiento SEO y los diseños receptivos se vuelven más sólidos. Además, puede disminuir el tamaño de los archivos y aumentar el rendimiento en general.

No obstante, una web no se vuelve accesible sólo por estar construida bajo una estructura semántica, también necesita de, atributos, propiedades y/o metadatos que mejoren el acceso a los contenidos.

Los datos personalizados son un tipo de atributos que suelen utilizarse para guardar datos privados en las páginas, no obstante, también sirven para asignar descriptores como es el caso de la WAI-ARIA.

Los metadatos y los elementos de cabecera, como puedan ser H1...H6 pueden ser también de gran ayuda en lo referente a mejorar la accesibilidad web, así como la integración con otras tecnologías como JavaScript, CSS o SMIL.

CSS (Cascading Style Sheets)

El lenguaje CSS es un lenguaje de diseño que permite la personalización de documentos estructurados escritos con otro lenguaje de marcado, como pueda ser HTML o XHTML.

El uso de CSS debe intentar utilizarse para contenidos que no sean relevantes, ni tampoco como elemento diferenciador de accesibilidad. Lo que sí se puede hacer es apoyarse en él para aumentar ayudar o aumentar la accesibilidad. Por ejemplo, un contenido no textual decorativo, como pueda ser una imagen de fondo, debe ser expuesta a través de CSS.

No obstante, también tiene otras cosas interesantes, como es el módulo de discurso o **CSS Speech Module**. Este complemento de CSS permite definir cómo se hablan o pronuncian los elementos de un documento.

Entre otras cosas, permite definir el volumen y distribución espacial de la voz, cómo se debe realizar la descripción auditiva del contenido de voz, dónde, cuándo y de cuanto deben ser los silencios o las pausas antes o después de los elementos, dónde, cuándo y qué sonidos se deben reproducir antes o después de lo elementos, el énfasis, velocidad, tipo y género de la voz y los estilos en elementos de tipo lista y contador.

JavaScript

Cuando se desea realizar una web accesible se debe tratar de no abusar del JavaScript porque puede bajar el rendimiento del sistema o interfaz. Además, no debe ser intrusivo, es decir, las funcionalidades de la página deben seguir funcionando, aunque el usuario decida desactivar la interpretación del código JavaScript.

También es importante que las acciones y eventos no se ejecuten por sí solas, es decir, no se deben mostrar diálogos emergentes, anuncios, llamadas a servidor, etc., si no el usuario no lo ha solicitado de manera expresa.

Si la ejecución de una acción implica la apertura de una ventana emergente o nueva, se debe informar previamente al usuario.

Y, cómo no, al igual que sucede con otros lenguajes como HTML y CSS, debe estar separado del resto, es decir, el CSS debe estar en un archivo diferente a los de HTML y JavaScript.

Flash

Si se decide utilizar esta tecnología, que personalmente no la recomiendo porque existen otras opciones mejores, se debe proporcionar equivalencias textuales siempre que se pueda, un contexto de la estructura de la película, en un orden correcto,

con todos los controles de animación visibles y proporcionar acceso por teclado a todos los controles que puedan ser manipulados por el dispositivo de puntero

Además, se deben utilizar todos los controles o funciones relacionados con la accesibilidad web, esto es, se deben utilizar los componentes de simple button, check box, radio button, label, text input, text area, combo box, list box, window, alert y data grid.

También es importante que se proporcionen subtítulos cuando se usen vídeos o audios, que se pueda controlar el vídeo o audio sin que interfiera con los asistentes de voz y, en general, dar soporte a los usuarios con discapacidad total o parcial de visión.

PDF (Portable Document Format)

Un PDF es accesible si el contenido puede ser utilizado por usuarios con o sin discapacidad e independientemente del contexto de uso.

Básicamente, para que un PDF sea accesible, se debe indicar el idioma del archivo, incluir textos alternativos a las imágenes informativas, proporcionar un etiquetado de todos los elementos del documento y asignarle todos los metadatos necesarios para que las tecnologías de asistencia puedan describirlo adecuadamente.

Además, es importante revisar el orden de lectura y la paginación, incluir textos alternativos a todos los enlaces describiendo su objetivo y contexto, asegurarse de que la secuencia de tabulación tiene el orden correcto y que los ajustes de seguridad no interfieran el acceso a la información que debe poder acceder el lector de pantalla o tecnología de asistencia.

XSL (Extensible Stylesheet Language)

XSL es un conjunto de recomendaciones que se utilizan para definir y mantener la transformación, presentación e interacción de información estructurada, sobre todo, en documentos XML.

Desde que se estandarizó HTML5 y se abandonaron los desarrollos en HTML estricto y transicional, cada vez menos se recurre a este tipo de tecnologías, a no ser que se esté trabajando en arquitecturas requerimientos muy específicos porque JSON es más ligero, rápido y personalizable.

Para mejorar la accesibilidad XSL permite presentar información visual y no visual con pretensiones de ayudar a CSS posibilitando funcionalidades no definidas a través de CSS, como pueda ser la reordenación de elementos.

Reproducción multimedia

Uno de los problemas que tienen los y videos y el multimedia es que no todos los navegadores soportan la reproducción a pantalla completa y, además, tener que personalizarlos de forma corporativa o sofisticada puede volverse una tarea muy ardua y tediosa.

Históricamente, los desarrolladores sólo podían incrustar un archivo que no tenía la posibilidad de reproducirse sin descargarlo por completo, adquirir un desarrollo de terceros (que normalmente no era compatible con todos los navegadores) o utilizar un servidor de medios dedicado (lo que suponía un incremento muy alto de mantenimiento).

Actualmente varias hay opciones para poder realizar streaming desde las interfaces, aunque según qué navegador vaya a reproducir el contenido multimedia, requerirá utilizar uno u otro tipo de codificación diferente.

En lo referente al tema que nos ocupa, al igual que pasa con las animaciones, sliders u otros componentes como son los banners, todos ellos pueden implementar a través de HTML5 y CSS. Eso sí, su implementación debe cumplir con todos los requerimientos descritos en la recomendación WCAG 2.1 comentada anteriormente.

SVG (Scalable Vector Graphics)

Los Gráficos Vectoriales Escalables o SVG son una forma de crear gráficos más accesibles, rápidos y efectivos. Se podría decir que los principales usuarios que se benefician son sólo los que presentan algún tipo de discapacidad o incapacidad, o aquellos usuarios que disponen de dispositivos y conexiones lentas, sin embargo, nada más lejos.

Por supuesto que beneficia, y mucho, a aquellas personas que presentan una discapacidad visual total o parcial o a los usuarios que usan tecnologías asistivas, pero también los demás usuarios se benefician porque reducen de manera considerable el tamaño de las páginas y transferencias, requieren menos recursos de memoria y CPU y permiten ser mostradas en cualquier resolución sin perder calidad.

La forma de hacer que un gráfico vectorial sea más accesible es proporcionar textos descriptivos en los objetos que indiquen su función, proveer a los controles de cualidades únicas que no se basen únicamente en el color, no incluir texto como paths o imágenes y no utilizar el elemento “g” o descripciones lógicas para cosas que no sean estructurar los documentos.

Además, es recomendable que se utilicen altos contrastes, medidas relativas y, si procede, se representen las relaciones matemáticas con algún lenguaje de marcado matemático como es **MathML**.

Silverlight

Microsoft Silverlight es una herramienta para aplicaciones web del mismo modo que lo hace Adobe Flash. Entre otras cosas, agrega funcionalidades multimedia como la reproducción de vídeos, gráficos vectoriales, animaciones e interactividad.

Aunque es una tecnología que ya no se utiliza en gran medida, las Pautas de Accesibilidad para el Contenido Web contemplan varias casuísticas y proporcionan muchos ejemplos para ayudar a solucionar todos los posibles problemas.

1.3 LEGISLACIÓN Y ESTÁNDARES

El Portal de la Administración Pública (PAe) pone a disposición de todos los ciudadanos, organismos, empresas y organizaciones el acceso y descarga de todas las normativas y legislación aplicables a la geografía española. La presente redacción es está extraída a febrero de 2020.

1.3.1 Norma EN 301 549:2018

Más información en

https://www.etsi.org/deliver/etsi_en/301500_301599/301549/02.01.02_60/en_301549v020102p.pdf

La norma EN 301 549:2018, titulada Requisitos de accesibilidad para productos y servicios TIC, actualmente en la versión 2.1.2 especifica los requisitos funcionales de accesibilidad aplicables a los productos y servicios que incluyan TIC (sitios web, software, apps nativas, documentos, hardware, etcétera). Además de describir los procedimientos de prueba y la metodología de evaluación a seguir para cada requisito de accesibilidad.

En esta nueva versión, declarada por la Comisión Europea como estándar armonizado para la aplicación de la Directiva de Accesibilidad Web en la Decisión de Ejecución (UE) 2018/2048 de la Comisión, debe ser aplicada desde el 21 de diciembre de 2018, para todas las Administraciones Públicas españolas.

1.3.2 Norma UNE 139803:2012

Más información en

http://administracionelectronica.gob.es/PAe/accesibilidad/UNE139803=2012.pdf

La UNE 139803:2012, titulada, Requisitos de Accesibilidad para contenidos en la web, es una norma que establece los requisitos referentes a las Pautas de Accesibilidad para el Contenido Web (WCAG), de la Iniciativa de Accesibilidad Web (WAI) y del Consorcio de la World Wide Web (W3C). Es equivalente a la WCAG 2.0 AA.

1.3.3 Estándar ISO/IEC 40500:2012

Más información en

http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=58625

El estándar ISO/IEC 40500:2012, titulada, Pautas de Accesibilidad para el Contenido Web (WCAG) 2.0, cubre una amplia gama de recomendaciones para hacer que el contenido web sea más accesible haciendo que personas con discapacidad auditiva, visual, física, intelectual o cognitiva puedan beneficiarse de Internet.

1.3.4 Comparativa de estándares sobre accesibilidad web

A continuación, se resumen algunas normas gubernamentales sobre los estándares de accesibilidad a nivel mundial. En general, estas normas se suelen aplicar a los sistemas de las agencias gubernamentales más que a sistemas comerciales,

a excepción de Australia y Noruega, donde todos los sistemas e interfaces deben cumplir la normativa.

País	*Estándar / Legislación*
Australia	WCAG 2 AA / **Disability Discrimination Act**
Canadá	WCAG 2 AA / Human Rights Act 1977
UE	WCAG 2 AA / European Parliament Resolution (2002)
Francia	**RGAA 3** (basado en WCAG 2) / **Law No 2005-102, Article 47**
Alemania	**BITV 2** (basado en WCAG 2) / Federal Disabled Equalization Law (BGG)
Hong Kong	WCAG 2 AA
India	GIGW (basado en WCAG 2 A) / **Guidelines for Indian Government Websites**
Irlanda	WCAG 2 AA / **The Disability Act 2005**
Israel	WCAG 2 AA / **Equal Rights of Persons with Disabilities Law, 5758-1998**
Italia	WCAG 2 /**Law No. 4/2004 (Stanca Law)**
Japón	**X 8341-3:2016** (iagual a WCAG 2)
Países Bajos	WCAG 2 AA
N. Zelanda	WCAG 2 AA / **Human Rights Amendment Act 2001**
Noruega	WCAG 2 AA (con excepciones) / **LOV 2008-06-20 nr 42**
Ontario	**AODA** (equivalente a WCAG 2 AA)
Quebec	SGQRI 008 (basado en WCAG 2) / **Standards sur l'accessibilité du Web**
España	WCAG 2 AA (UNE 301 549:2019)
Reino Unido	WCAG 2 AA / **Equality Act 2010**
USA	Section 508 (basado en WCAG 1) / **Section 508 of Rehabilitation Act**

Figura 1.1. Resumen de normas a nivel mundial.

1.3.5 Estándar SMIL

Más información en

http://www.w3.org/tr/smil3/

SMIL (Synchronized Multimedia Integration Language) es un estándar de la W3c que está basado en XML y que permite a los diseñadores integrar audio, video, imágenes, texto o cualquier otro contenido multimedia a las interfaces. Ahora mismo está vigente la versión 3.0.

Su antecesor, SMIL1.0 permitía a los desarrolladores o diseñadores describir el comportamiento temporal de la presentación, su disposición en la pantalla y asociar enlaces a los objetos.

Según se ha ido avanzando en número de versión, se han ido ganando nuevas características que han ayudado a la navegación y animación, que han proporcionado soporte para realizar broadcast, han incluido nuevas funcionalidades en el formato visual y un incremento en el rendimiento, característica de gran importancia si se contextualiza en el mundo de los dispositivos móviles.

Finalmente, en diciembre de 2008 aparece la recomendación SMIL 3.0 que se desarrolla pensando en la construcción de aplicaciones multimedia en plataformas que soportan los estándares Web. Por ejemplo, en este nuevo estándar se pueden añadir presentaciones multimedia de forma segura a otras aplicaciones XML, incluyendo HTML y SVG. SMIL 3.0, además, facilita el desarrollo de aplicaciones multimedia sobre plataformas móviles y posee una versión llamada "SMIL Tiny" que es un perfil mínimo de SMIL 3.0 perfecto para sistemas incrustados y aplicaciones ligeras, como reproductores multimedia.

SMIL 3.0 es un estándar que beneficia a todos, pero especialmente, a los usuarios con discapacidad visual ya que permite cubrir sus necesidades de una forma sencilla y organizada.

1.3.5.1 MÓDULOS DE SMIL

- **Módulo de disposición**: Permite definir las propiedades o atributos para posicionar los contenidos, el orden de visualización en espacios coincidentes, tamaño y posición de las regiones, el color de fondo o la forma de ajuste.
- **Módulo de sincronización**: Permite definir las propiedades o atributos para establecer los valores de inicio y fin por defecto, la duración, el modo de reproducción, el tipo de iteración, número de veces a iterar y los valores máximo y mínimo del objeto multimedia, entre otros.
- **Módulo de animaciones**: Permite cambiar dinámicamente las propiedades de objetos de contenido como el color o posición, el modo de cambio y/o el tipo de cambio.
- **Módulos de control de contenidos**: Permite definir las propiedades o atributos para controlar la representación de uno u otro contenido

mediante son el bitrate, el idioma, el tamaño de la pantalla, los subtítulos y/o la CPU.

- **Módulo de enlaces**: Permiten definir algunas propiedades para interactuar con los usuarios.
- **Módulo de metadatos**: Permite definir la descripción del contenido como es el autor, el título o el email, por ejemplo.
- **Módulo de transiciones**: Permite definir cómo se van a realizar las transiciones en el objeto multimedia, su duración, color de desvanecimiento y el modo de transición, entre otros.

Existen bastantes reproductores que permiten leer e interpretar ficheros SMIL y trascribir las acciones que en él se describen.

1.3.5.2 EJEMPLO DE CONTENIDO SMIL

```
<smil>
    <head>
        <meta name="author" content="Pablo Fernández"/>
        <meta name="title" content="Ejemplo multimedia"/>
        <meta name="copyright" content="(c)2018 PEFC"/>
    </head>
   <body>
        <switch>
            <par system-bitrate="700000">
            <!—Para resoluciones >= 1280x720 -->
                <audio src="audio/audioHD.snd"/>
                <video src="video/videoHD.avi"/>
                <image src="lyrics/imagenHD.jpg"/>
            </par>
            <par system-bitrate="350000">
            <!—Para resoluciones >= 320x240 -->
                <audio src="audio/audioMobile.snd"/>
                <video src="video/videoMobile.avi"/>
                <image src="lyrics/imagenMobile.jpg"/>
            </par>
        </switch>
    </body>
</smil>
```

Código 8.1. Ejemplo de descripción multimedia SMIL.

1.3.6 La iniciativa WAI ARIA

Más información en

https://www.w3.org/WAI/standards-guidelines/aria/

La WAI ARIA (Web Accessibility Initiative Accessible Rich Internet Applications) es una iniciativa del W3C que define o describe una forma de realizar contenidos accesibles. Es muy eficiente con contenidos dinámicos y desarrollos creados bajo los lenguajes HTML, Ajax o JavaScript.

El objetivo principal de este estándar es proporcionar información adicional y útil en las diferentes partes del contenido, sirviendo de ayuda para los usuarios finales que utilizan tecnologías asistivas tales como un lector de pantalla.

La WAI ARIA proporciona una serie de atributos que funcionan como identificadores de las diferentes partes de la aplicación que interactúa con el usuario. También se incluyen mapeo de controles, roles y eventos para la accesibilidad de las APIs (Application Programming Interfaces).

1.3.6.1 PARTES DE LA WAI ARIA

WAI ARIA propone a los desarrolladores una serie de soluciones destinadas a **hacer accesibles widgets, áreas activas y demás componentes enriquecidos** que se encuentran en la mayoría de las aplicaciones web en la actualidad.

Para ello se describen unos **roles y propiedades** con la finalidad de otorgar de información a los productos de apoyo y para que interactúen adecuadamente con los componentes más normales de las aplicaciones web.

1.3.6.1.1 Roles

Los roles de WAI ARIA proporcionan un nombre que identifica la funcionalidad de la estructura o contenido. A continuación, se muestra el ejemplo de un widget sencillo en el que se ha querido representar una barra de herramientas con las tres funcionalidades propias del portapapeles.

```
<ul role="toolbar" tabindex="0" aria-activedescendant="copy">
    <li id="copy">Copiar</li>
```

```
    <li id="cut">Cortar</li>
    <li id="paste">Pegar</li>
</ul>
```

Código 8.2. Ejemplo de barra de navegación para gestión del portapapeles.

Otro ejemplo de muestra sobre el atributo role es el utilizado en HTML5 para formar estructuras usables y accesibles:

```
<nav id="nav" role="navigation">
    <!-- contenido de navegacion -->
</nav>
<section id="main" role="main">
    <!-- contenido principal -->
</section>
<div id="banner" role="banner">
    <!-- contenido anuncios -->
</div>
```

Código 8.3. Ejemplo de descripción de estructura.

1.3.6.1.2 Estados y propiedades

Las propiedades pueden establecer diferentes estados en los componentes, definir regiones dónde actualizar contenidos o describir las funciones de arrastrar y soltar.

A diferencia de los roles que sólo disponen de un atributo para definir los valores, los atributos de estados y propiedades son muchos y cada uno de ellos puede tomar uno o varios valores. Además, algunos de los estados y propiedades son aplicables de manera global a todos los elementos independientemente de si se aplica un rol o no.

```
<h1 id="title1">Vista de la Vía Láctea desde Ávila</h1>
<p id="description">
   <!-- contenido de la descripción -->
</p>
<picture>
    <img src="allocate1.png"
         alt="descripción corta"
         aria-labelledby="title1"
         aria-describedby="descripction" />
</picture>
```

Código 8.4. Ejemplo de descripción de una imagen accesible.

Atributos de componente

Estos están pensados para apoyar a los roles y para definir elementos de E/S. A continuación, se muestra un ejemplo típico de definición de atributos WAI ARIA para un campo de entrada de tipo texto.

```
<label for="name">Nombre
    <input type="text"
           id="name"
           name="name"
           required="required"
           aria-required="true" />
</label>
```

Código 8.5. Ejemplo de descripción de campo de texto accesible.

También se pueden establecer atributos que definan las regiones activas que pueden ser actualizadas aun sin hacerse a petición del usuario.

```
<p aria-live="polite">Nombre
    <!-- contenido del párrafo -->
</p>
```

Código 8.6. Ejemplo de definición de región activa.

Si observamos el código anterior, al elemento de párrafo se le ha añadido un atributo de región activa ARIA-LIVE con el valor POLITE. Una tecnología asistiva que reconozca el estándar WAI ARIA sabrá que párrafo será un área activa que podrá ser actualizada, en un futuro, con otro contenido.

Si el valor del atributo es POLITE, el contenido podrá ser actualizado una vez haya acabado las tareas que esté haciendo en ese momento el usuario. Si el valor del atributo es ASSERTIVE, el contenido podrá ser actualizado, aunque el usuario no haya terminado las tareas en ese momento.

1.3.6.1.3 Atributos de arrastrar y soltar

Los atributos de arrastrar y soltar (drag & drop) permiten proporcionar información de cómo se realiza la funcionalidad.

```
<div role="menuitem" aria-dropeffect="copy move">
    <!-- contenido del párrafo -->
</div>
```

Código 8.7. Ejemplo de descripción de efecto drag & drop.

En el ejemplo anterior, el rol con valor MENUITEM permite establecer la propiedad ARIA-DROPEFFECT para indicar que tipo de acción acepta el elemento.

1.3.6.1.4 Atributos de relaciones

En ocasiones no se pueden establecer, a partir de las estructuras del documento, las relaciones o pertenencias de los elementos que lo forman. Para estas situaciones, el estándar WAI ARIA permite definir las dependencias a través de atributos relacionales. A continuación, se muestra un ejemplo:

Como se ha visto en el código 11.4, se ha descrito un contenido visual a través de los atributos ARIA-DESCRIBEDBY y ARIA-LABELLEDBY del elemento que contiene la imagen. El atributo ARIA-LABELLEDBY ayuda a identificar el contexto y el atributo ARIA-DESCRIBEDBY proporciona la descripción asociado a ese contexto.

1.3.6.1.5 Acceso mediante teclado

Cuando se habla de accesibilidad es muy frecuente sacar el tema de los atajos de teclado. Siempre que sea posible se deben establecer atajos de teclado. De esta manera, se podrá dar soporte a las personas que no dispongan de ratón, por ejemplo.

En HTML 4, sólo los enlaces, campos de formulario, objetos, áreas y botones podían tomar el foco. En HTML5, todos los elementos pueden adquirir el foco gestionando el atributo TABINDEX que permite establecer un orden específico de navegación. Su valor por defecto u omisión es 0 y significa que la navegación se realizará en el orden de aparición en el documento. Si el valor se establece a -1, el elemento no podrá ser objeto del foco y, por lo tanto, se saltará.

La WAI ARIA, además, permite especificar otros comportamientos asociados a los hijos de los componentes que enriquecen el documento.

1.3.6.2 SOPORTE EN NAVEGADORES Y PRODUCTOS DE APOYO

El DOM (Document Object Model) contiene la estructura jerárquica y semántica del documento y, uno de sus usos, es para generar componentes propios de las aplicaciones enriquecidas.

Las tecnologías asistivas pueden utilizar el DOM para identificar los objetos, sin embargo, cuanta más información se proporcione a estas aplicaciones, mejor experiencia de usuario tendrá.

Las API de accesibilidad proporcionan los roles, estados, atributos, etcétera para que puedan ser utilizadas por las tecnologías asistiva, como lectores de pantalla. Cuando se utiliza WAI ARIA, la semántica proporcionada debe estar acorde a los

valores que se establecen en estas API para obtener un funcionamiento óptimo de las tecnologías asistivas.

La W3C proporciona un documento técnico dónde explica con detalle la asignación de las diferentes características de WAI ARIA con las Accessibility API más comunes. http://www.w3.org/TR/wai-aria-implementation/.

Hoy en día, prácticamente existen muchos productos que soportan la implementación de WAI ARIA, incluyendo navegadores, productos de apoyo y otras herramientas de desarrollo.

1.3.6.3 PRINCIPALES ATRIBUTOS DE LA WAI-ARIA

La accesibilidad web es algo sumamente importante y sumamente difícil de aplicar si no se entiende bien lo que se desea hacer. Por ello, antes de nada, vamos a precisar las principales propiedades con las que se puede conseguir proporcionar accesibilidad en los sitios y páginas web.

1.3.6.3.1 Atributo role

Los roles son unos atributos que se establecen para indicar la función u objetivo del elemento. Esto se vuelve necesario porque, en ocasiones, no es fácil discernir la diferencia u objeto del elemento mostrado. Por ejemplo, no es lo mismo un elemento que representa a una barra de progreso, que un elemento que representa a una barra de carga en proceso.

Los roles pueden ser de dos tipos. De tipo interfaz, que son lo que representan a elementos como árboles, listas, sliders, diálogos emergentes, etcétera y, de tipo estructural, que son los que representan o definen una estructura como pueda ser un menú de navegación o una cabecera o pie de página.

Aunque hemos dicho que existen dos tipos de roles, cabe destacar que, dentro de los roles estructurales, podemos encontrar un tercer tipo que se utiliza para diferenciar las diferentes secciones dentro de la estructura. Este tipo de roles, se suelen denominar LANDMARK ROLES y son los siguientes:

- **APPLICATION**: Indica que la región es una aplicación web, en vez de un documento web.
- **BANNER**: Indica que la región o sección contiene el título principal o el título interno de la página. Este rol suele estar asociado al elemento HEADER que contiene la cabecera de página.
- **COMPLEMENTARY**: Indica que es una sección que tiene contenido principal, pero es independiente y significativa por sí sola. Este rol suele estar asociado al elemento ASIDE que contiene la zona lateral o anexa al contenido principal.
- **CONTENTINFO**: Indica que la sección contiene información relevante sobre el documento principal, como derechos de autor o enlaces a declaraciones de privacidad. Este rol suele estar asociado al elemento FOOTER que contiene el pie de página.
- **FORM**: Indica que la sección representa una colección de elementos de formulario, sean editables o no.
- **MAIN**: Indica la sección de contenido principal en un documento. Aunque puede darse el caso de que no, por norma general, una página tendrá una única región establecida a este valor. Este rol suele estar asociado al elemento MAIN que contiene el contenido principal del documento.
- **NAVIGATION**: Indica que la sección contiene una colección de enlaces o acciones pensados para navegar por el sitio web. Este rol suele estar asociado al elemento NAV que contiene el menú principal de navegación y/o menús secundarios de navegación o enlaces.
- **SEARCH**: Indica que la sección o elemento tiene la función de buscador para el sitio web. Este rol suele estar asociado a elementos de bloque, como pueda ser un DIV y que suelen contener un elemento INPUT de tipo SEARCH.

Ejemplos:

```
<header id="header" role="banner">...</header>
<div id="sitelookup" role="search">...</div>
<nav id="nav" role="navigation">...</nav>
<main id="content" role="main">...</main>
<aside id="rightsideadvert" role="complementary">...</aside>
<footer id="footer" role="contentinfo">...</footer>
```

No obstante, como decíamos, existen otros roles que no son especiales y que suelen y deben asignarse a los botones, enlaces, elementos de formulario, iconos y regiones de manera que proporcionen una cantidad de información suficiente sobre lo que representa el elemento.

Entre ellos podemos encontrar:

- **ALERT**: Indica que el elemento representa un mensaje que contiene información importante que, por lo general, es urgente. Cabe destacar que, cuando este rol se establece, también suele requerir la especificación de los atributos ARIA-LIVE y ARIA-ATOMIC.
- **ALERTDIALOG**: Indica que el elemento representa un diálogo que contiene información, pero no requiere de una interacción con el usuario.
- **ARTICLE**: Indica que el elemento representa una acción que se activa por pulsación o activación del usuario.
- **BUTTON**: Indica que el elemento representa un contenido independiente de un documento, página o sitio web. Cabe destacar que, cuando este rol se establece, también suele requerir la especificación de los atributos ARIA-HASPOPUP, ARIA-PRESSED y ARIA-DISABLED.
- **CHECKED**: Indica que el elemento lleva implícito un estado de verificación que puede ser verdadero, falso o mixto. Cabe destacar que, cuando este rol se establece, también suele requerir la especificación del atributo ARIA-CHECKED.
- **COLUMNHEADER**: Indica que el elemento contiene información de encabezado de una columna.
- **COMBOBOX**: Indica que el elemento representa un desplegable, cuadro de texto donde los usuarios pueden escribir con anticipación para seleccionar una opción, o escribir un texto cualquiera que se toma como elemento nuevo de lista. Cabe destacar que, cuando este rol se establece, también suele requerir la especificación de los atributos ARIA-HASPOPUP y ARIA-EXPANDED.
- **COMMAND**: Indica que el elemento representa una acción, pero no recibe datos de entrada.
- **COMMAND**: Indica que el elemento puede contener descendientes navegables o propios.
- **DEFINITION**: Indica que el elemento es una definición o concepto.
- **DIALOG**: Indica que el elemento representa un cuadro de diálogo que lleva asociada una interacción que solicita información o una respuesta. Cabe destacar que, cuando este rol se establece, también suele requerir la especificación del atributo ARIA-HASPOPUP.
- **DIRECTORY**: Indica que el elemento contiene una tabla o listado. Es decir, que representa una lista de referencias a miembros de un grupo, como una tabla de contenido estática.

- **DOCUMENT**: Indica que el elemento contiene información relacionada que se declara como contenido de documento y no como una aplicación web.
- **GRID**: Indica que el elemento representa una colección de elementos (o celdas) que están organizados a modo de filas y columnas, como si de una tabla se tratase.
- **GRIDCELL**: Indica que el elemento es uno de los elementos o celda que contiene el elemento padre que tiene el ROLE de GRID.
- **GROUP**: Indica que el elemento representa una colección de elementos de interfaz de usuario que no están incluidos en el resumen de la página o tabla de contenido por las herramientas de asistencia.
- **INPUT**: Indica que el elemento funciona como un componente de entrada de datos del usuario.
- **LINK**: Indica que el elemento es un enlace o vínculo a un recurso, externo o interno, que provoca la navegación hasta ese destino.
- **LIST**: Indica que el elemento representa una lista de elementos que permiten la interacción con el usuario.
- **LISTBOX**: Indica que el elemento representa una lista de elementos que sólo permite la selección de uno de sus elementos.
- **LISTITEM**: Indica que el elemento es uno de los elementos contenidos por el elemento padre que tiene el ROLE de LIST o LISTBOX. Cabe destacar que, cuando este rol se establece, también suele requerir la especificación del atributo ARIA-LEVEL
- **MENU**: Indica que la sección es un widget que contiene una lista de opciones para el usuario.
- **MENUBAR**: Indica que el elemento es un subelemento del elemento padre que tiene el ROLE de MENU. Habitualmente se identifica con una barra horizontal que, además, suele estar visible.
- **MENUITEM**: Indica que el elemento es una de las opciones ofrecidas por el elemento padre que tiene el ROLE de MENU o MENUBAR.
- **MENUITEMCHECKBOX**: Indica que el elemento es una de las opciones ofrecidas por el elemento padre que tiene el ROLE de MENU o MENUBAR. No obstante, este elemento lleva implícito un estado de verificación que puede ser falso, verdadero o mixto. Cabe destacar que, cuando este rol se establece, también suele requerir la especificación del atributo ARIA-CHECKED.

- **MENUITEMRADIO**: Indica que el elemento es una de las opciones ofrecidas por el elemento padre que tiene el ROLE de MENU o MENUBAR. No obstante, este elemento lleva implícito un estado de selección que puede ser verdadero o falso, pero con la diferencia de que, solamente, permite un único elemento seleccionado por grupo. Cabe destacar que, cuando este rol se establece, se debe comprobar que no existan más dentro de la misma subsección o del mismo grupo. Si fuese necesario, los diferentes elementos de MENUITEMRADIO pueden separarse a través de subelementos padre con el ROLE establecido a GROUP. Si existen varios subelementos con el ROLE establecido a GROUP, se deben separar mediante elementos que tengan el ROLE establecido a SEPARATOR.
- **PROGRESSBAR**: Indica que el elemento muestra el estado de progreso de una tarea o acción que consta de varios pasos o tiene predefinido que va a tardar bastante tiempo en completarse. Cabe destacar que, cuando este rol se establece, también suele requerir la especificación de los atributos ARIA-VALUEMIN, ARIA-VALUEMAX y ARIA-VALUENOW.
- **ROW**: Indica que el elemento representa a una de las filas de una colección de elementos en forma de cuadrícula, como pueda ser una tabla.
- **ROWGROUP**: Indica que el elemento contiene un grupo de elementos de fila de una colección de elementos en forma de cuadrícula, como pueda ser una tabla.
- **ROWHEADER**: Indica que el elemento contiene información de encabezado de una fila asociada con una colección de elementos en modo tabla o cuadrícula.
- **SEPARATOR**: Indica que el elemento funciona como separador o divisor de regiones de acciones agrupadas por el rol GROUP.
- **TAB**: Indica que el elemento es una de las pestañas ofrecidas por el elemento padre que tiene el ROLE de TABLIST.
- **TABLIST**: Indica que el elemento es un widget que representa una funcionalidad por pestañas.
- **TABPANEL**: Indica que el elemento contiene los recursos y elementos asociados a una pestaña y, por tanto, a uno de los elementos que tiene el rol establecido a TAB.
- **TOOLTIP**: Indica que el elemento es un widget que representa una funcionalidad de árbol.
- **TREE**: Indica que el elemento es un widget que representa una funcionalidad de árbol.

- **TREEITEM**: Indica que el elemento es una de las ramas ofrecidas o listadas por el elemento padre que tiene el ROLE de TREE. Cabe destacar que, cuando este rol se establece, también suele requerir la especificación del atributo ARIA-LEVEL.

Ejemplos:

```
<ul role="tablist">
   <li role="tab">Pestaña 1</li>
   <li role="tab">Pestaña 2</li>
</ul>
<div role="progressbar">
<select role="listbox">
   <option role="listitem" aria-level="1">Opción 1</option>
   <option role="listitem" aria-level="1">Opción 2</option>
</select>
```

Cabe destacar que, existen muchos más roles que los aquí presentados, aunque sí se podría decir que están los más utilizados. Si se desean ver todos los roles o, simplemente, se desea más información sobre uno de ellos, se puede visitar la URL o dirección web *https://www.w3.org/WAI/PF/aria-1.1/roles*.

1.3.6.3.2 Atributos aria-autocomplete y aria-activedescendant

El atributo ARIA-AUTOCOMPLETE permite indicar, a las herramientas de asistencia, que el elemento conlleva una búsqueda predictiva con una posterior visualización de resultados total o parcial. Además, permite establecer el tipo de interacción que está asociado al elemento de formulario.

Los posibles valores que puede tomar el atributo ARIA-AUTOCOMPLETE son el valor INLINE, para indicar que el valor resultante de la búsqueda predictiva se mostrará dentro del elemento de formulario al que está asociado, LIST, para indicar que el resultado de la búsqueda predictiva se mostrará en un elemento de lista a parte o BOTH, para indicar que el elemento de formulario ofrece ambos modelos al mismo tiempo.

Ejemplo:

```
<input id="cb1-edit"
       type="text"
       aria-activedescendant="opt04"
       aria-owns="resultados-busqueda"
       aria-autocomplete="list"
       role="combobox" />
```

```
<ul aria-expanded="true" role="listbox" id="resultados-busqueda">
    <li role="option" id="opt01">HTML para todos</li>
    <li role="option" id="opt02">La guía oficial de HTML5</li>
    <li role="option" id="opt03">Creación de páginas con HTML5</li>
    <li role="option" id="opt04">Accesibilidad Web y HTML5</li>
    <li role="option" id="opt05">La usabilidad de HTML5</li>
</ul>
```

Como puede apreciarse, el atributo ARIA-ACTIVEDESCENDANT permite establecer el elemento de la lista que, actualmente, está seleccionado.

1.3.6.3.3 Atributo aria-atomic

El atributo ARIA-ATOMIC permite establecer si la actualización de un contenedor afecta a todas o sólo a algunas partes. Esta actualización será revelada en función de las notificaciones de cambio definidas por el atributo ARIA-RELEVANT.

Ejemplo:

```
<h3>Contenido del carrito</h3>

<div aria-live="polite" aria-atomic="true">
    <div>Su cesta de la compra contiene
        <span id="nArt">0</span>
        Artículos
    </div>
</div>
```

1.3.6.3.4 Atributo aria-checked

El atributo ARIA-CHECKED permite establecer el estado de aquellos elementos de formulario que resultan ser de tipo casilla de verificación o de tipo radio. Básicamente, el atributo ARIA-CHECKED es idéntico al atributo CHECKED de HTML, salvo por la diferencia de que ARIA-CHECKED permite establecer un estado adicional que indica no es ni activado, ni desactivado.

Los posibles valores que puede tomar el atributo ARIA-CHECKED son FALSE, para indicar que no está verificado o seleccionado, TRUE, para indicar que está verificado o seleccionado y MIXED, para indicar que la verificación o selección es sólo parcial.

Al margen de los valores de estado TRUE y FALSE, el valor de estado MIXED puede ser útil cuando el estado de la casilla de verificación hace referencia a un conjunto de elementos de su mismo tipo en donde hay algunos seleccionados y otros que no.

Ejemplo:

```
<table>
    <thead>
        <tr>
            <th>
                <input type="checkbox" id="checkAll" aria-checked="mixed"/>
            </th>
            <!-- ... otros elementos TH -->
        </tr>
    </thead>
    <tbody>
        <tr>
            <td>
                <input type="checkbox" id="chk01" aria-checked="true"/>
            </td>
            <!-- ... otros elementos TD -->
        </tr>
        <tr>
            <td>
                <input type="checkbox" id="chk02" aria-checked="false"/>
            </td>
            <!-- ... otros elementos TD -->
        </tr>
        <!-- ... otros elementos TR -->
    </tbody>
```

1.3.6.3.5 Atributo aria-controls

El atributo ARIA-CONTROLS permite establecer una relación de pertenencia que indica que elementos puede controlar. Esto se hace necesario cuando esa relación no viene expresada o no está representada en el DOM.

Por ejemplo, un botón que abre una serie de opciones a modo de menú podría definirse de la siguiente manera:

Ejemplo:

```
<div class="menu-layer">
  <button id="menuBtn" aria-haspopup="true" aria-controls="submenu">
      Ver opciones
  </button>
  <ul id="submenu" role="menu" aria-labelledby="menuBtn">
    <li role="none">
        <a role="menuitem" href="#">
            Opción 1
```

```
        </a>
    </li>
    <li role="none">
        <a role="menuitem" href="#">
            Opción 2
        </a>
    </li>
</ul>
```

Como se puede apreciar en el código anterior, el botón establece una relación con el elemento de lista a través de su atributo ID. La lista, a su vez, establece una relación con el elemento botón a través del atributo ARIA-LABELLEDBY.

Al pulsar en el botón, el sistema podría realizar la actualización del atributo ARIA-EXPANDED en el elemento a través de JavaScript y, estableciéndolo a TRUE si está mostrando el elemento UL, o a FALSE si está oculto.

1.3.6.3.6 Atributo aria-describedby

El atributo ARIA-DESCRIBEDBY permite establecer una descripción larga o detallada para todos aquellos elementos mediante el establecimiento del valor de un ID válido. Este identificador debe ser un valor válido definidos por un atributo ID alcanzables en el mimo contexto. Es decir, no se pueden utilizar valores de ID que estén ubicados o definidos en otra ventana o frame diferente a la actual.

Esto suele ser útil en situaciones donde el contenido a mostrar es un enlace a un archivo descargable (como un PDF o Excel) o cuando el elemento al que hace referencia no está dentro del mismo contenedor.

Ejemplo:

```
<form role="form"
    <label for="username">Nombre de Usuario</label>
    <input type="text" id=" usr " name="usr" required />

    <label for="username">Contraseña</label>
    <input type="password" id="pwd" name="pwd" required
           aria-describedby="pwdDesc" />

    <p id="pwdDesc">La contraseña debe tener, cómo mínimo, una mayúscula, una
minúscula, un número y un caracter especial. Además, no puede tener una longitud
menor a 6 caracteres.</p>

    <button type="submit" class="accept rounded">Acceder</button>
</form>
```

1.3.6.3.7 Atributo aria-describedat

El atributo ARIA-DESCRIBEDAT tiene un comportamiento idéntico al atributo ARIA-DESCRIBEDBY, si exceptuamos que lo que se establece es una URL, en vez de un ID.

Esto suele ser útil cuando la descripción no es una frase, sino una descripción larga o representa una explicación que conlleva varios párrafos.

Ejemplo:

```
<a href="#"
   aria-describedat="https://es.wikipedia.org/wiki/Magnitud_aparente">
        Magnitud aparente
</a>
```

1.3.6.3.8 Atributo aria-disabled

El atributo ARIA-DISABLED permite indicar, a las herramientas de asistencia, que el elemento está desactivado o deshabilitado. Básicamente, es lo mismo que el atributo DISABLED de HTML, sin embargo, ARIA-DISABLED puede ser útil cuando no se desea realizar la validación nativa del navegador o herramienta de asistencia.

En general, se suele establecer en tiempo de ejecución, tras un proceso de validación en JavaScript.

Ejemplo:

```
<label for="nombre">Nombre:</label>
<input id="nombre" type="text" disabled aria-disabled="true" />
```

1.3.6.3.9 Atributo aria-expanded

El atributo ARIA-EXPANDED permite establecer si el elemento que representa a un contenido plegable o colapsable como pueda ser un menú desplegable de navegación, está expandido o contraído.

Por ejemplo, un menú lateral deslizante, también conocido como Off-Screen Menu, con un botón tipo hamburguesa podría definirse de la siguiente manera:

Ejemplo:

```
<button class="menu-toggle"
        aria-label="Menú principal"
        aria-expanded="false">
    <i class="icon bars">□</i>
</button>
```

```
<aside class="hidden" role="navigation">
    Ejemplo de menú
    <ul>
        <li><a href="./home.html">Home / Inicio</a></li>
        <li><a href="./quienes-somos.html">Quienes Somos</a></li>
        <li><a href="./servicios.html">Servicios</a></li>
        <li><a href="./donde-estamos.html">Dónde estamos</a></li>
    </ul>
</aside>
```

Al pulsar en el botón, el sistema podría realizar la actualización del atributo ARIA-EXPANDED en el elemento a través de JavaScript y, estableciéndolo a TRUE si está mostrando el ASIDE, o a FALSE si está oculto.

1.3.6.3.10 Atributo aria-flowto

El atributo ARIA-FLOWTO permite cambiar o alterar el orden normal de lectura proporcionado por el documento y pasar al elemento cuyo identificador es el valor indicado. No obstante, cuando este atributo presenta múltiples valores, se tomarán como opciones o alternativas para el siguiente contenido en el orden de lectura.

Aunque parezca algo evidente de mencionar, todas las propiedades de ARIA, incluyendo la propiedad ARIA-FLOWTO, serán ignoradas por el navegador. Recordemos que el navegador no enriende de lecturas alternativas e implementa el orden de tabulación a través de métodos como el atributo TABINDEX.

Por ejemplo, cambiar el orden de lectura en las herramientas de asistencia, para que lean primero la noticia de en medio, luego la primera y, por último, la tercera, podría definirse de la siguiente manera:

Ejemplo:

```
<h1 aria-flowto="no1">Noticias</h1>

<h2>Tiempo</h2>
<div id="no2" title="Tiempo en Madrid" aria-flowto="no3">
    <p>El tiempo será estable durante las próximas horas, ...</p>
</div>

<h2>Covid-19</h2>
<div id="no1" title="Actualidad sobre el Covid-19" aria-flowto="no2">
    <p>Se proporcionan las primeras vacunas a los españoles</p>
</div>
```

```
<h2>Presupuestos Generales del Estado (PGE)</h2>
<div id="no3" title="Ultimas novedades sobre los PGE">
    <p> Los PGE para 2021 presentan algunos cambios de ... </p>
</div>
```

1.3.6.3.11 Atributo aria-haspopup

El atributo ARIA-HASPOPUP permite establecer si el elemento abre un menú o un desplegable de acciones.

Por ejemplo, un botón que abre una serie de opciones a modo de menú podría definirse de la siguiente manera:

Ejemplo:

```
<div class="menu-layer">
  <button id="menuBtn" aria-haspopup="true" aria-controls="submenu">
      Ver opciones
  </button>
  <ul id="submenu" role="menu" aria-labelledby="menuBtn">
    <li role="none">
        <a role="menuitem" href="#">
            Opción 1
        </a>
    </li>
    <li role="none">
        <a role="menuitem" href="#">
            Opción 2
        </a>
    </li>
</ul>
```

Al pulsar en el botón, el sistema podría realizar la actualización del atributo ARIA-EXPANDED en el elemento a través de JavaScript y, estableciéndolo a TRUE si está mostrando el elemento UL, o a FALSE si está oculto.

1.3.6.3.12 Atributo aria-hidden

El atributo ARIA-HIDDEN permite establecer si el elemento está oculto, o por el contrario, esta visible. Si el atributo no está presente, o el valor de ARIA-HIDDEN está establecido a FALSE, se entenderá como que está visible.

Por ejemplo, un botón que abre una serie de opciones a modo de menú podría definirse de la siguiente manera:

Ejemplo:

```
<div class="menu-layer">
  <button id="menuBtn" aria-haspopup="true" aria-controls="submenu">
      Ver opciones
  </button>
  <ul id="submenu" role="menu" aria-labelledby="menuBtn" aria-hidden="true">
    <li role="none">
        <a role="menuitem" href="#">
            Opción 1
        </a>
    </li>
    <li role="none">
        <a role="menuitem" href="#">
            Opción 2
        </a>
    </li>
</ul>
```

Al pulsar en el botón, el sistema podría realizar la actualización de los atributos ARIA-HIDDEN y ARIA-EXPANDED en el elemento a través de JavaScript. Si el elemento se encuentra expandido, es decir, visible, podría establecerse el atributo ARIA-EXPANDED a TRUE y el atributo ARIA-HIDDEN a FALSE. En cualquier otro caso, se podría establecer el atributo ARIA-EXPANDED a FALSE y el atributo ARIA-HIDDEN a TRUE.

1.3.6.3.13 Atributo aria-invalid

El atributo ARIA-INVALID permite indicar, a las herramientas de asistencia, que el elemento de formulario es un campo de entrada no cumple con las expectativas o formato indicados.

Esto es útil cuando se están solicitando datos preformateados como puedan ser los correos electrónicos, teléfonos o cualquier otro tipo de entrada que pueda responder a una posible máscara de entrada, pero también es útil para indicar que no está relleno y que, por tanto, es obligatorio.

Los posibles valores que puede tomar el atributo ARIA-INVALID son FALSE, para indicar que no se detectaron errores, GRAMMAR, para indicar que se detectó un error gramatical, SPELLING, para indicar que se detectó un error ortográfico o, simplemente, TRUE, para indicar que el proceso de validación no fue superado.

En general, se suele establecer en tiempo de ejecución, tras un proceso de validación.

Ejemplo:

```
<label for="nombre">Nombre completo:</label>
<input id="nombre" type="text"
       aria-required="true"
       aria-invalid="false"
       oninput="checkValidity(this)" />

<script>
    function checkValidity(el){
        var invalid = (el.value.trim().length == 0);
        if (invalid) {
            el.setAttribute("aria-invalid", "true");
        } else {
            el.setAttribute("aria-invalid", "false");
        }
    }
</script>
```

1.3.6.3.14 Atributo aria-label

El atributo ARIA-LABEL permite establecer un nombre accesible a aquellos elementos que no poseen una descripción textual en su contenido.

Como se vio en la práctica del capítulo de Usabilidad Web, es frecuente utilizarlo cuando los elementos representan una imagen o icono, pero también es usable en cualquier elemento que requiera una descripción adicional que permita contextualizar y dar un significado inequívoco.

Ejemplo:

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

1.3.6.3.15 Atributo aria-labelledby

El atributo ARIA-LABELLEDBY permite establecer unos identificadores que serán utilizados para generar una descripción accesible. Estos identificadores deben ser valores válidos definidos por atributos ID alcanzables en el mimo contexto. Es decir, no se pueden utilizar valores de ID que estén ubicados o definidos en otra ventana o frame diferente a la actual.

Esto suele ser útil en situaciones donde el contenido a mostrar es un enlace a un archivo descargable (como un PDF o Excel) o cuando el elemento al que hace referencia no está dentro del mismo contenedor.

Ejemplo:

```
<p id="tInforme">Exportar informe en:</p>
<div class="export">
    <a href="./mayo.pdf" id="pdf" aria-labelledby="tInforme pdf">PDF</a>
    <a href="./mayo.xls" id="xls" aria-labelledby="tInforme xls">Excel</a>
</div>
```

1.3.6.3.16 Atributo aria-level

El atributo ARIA-LEVEL permite establecer el nivel jerárquico dentro de una determinada estructura.

Por ejemplo, una lista de opciones de menú con varios niveles de interacción podría definirse de la siguiente manera:

Ejemplo:

```
<ul role="list">
    <li role="listitem" aria-level="1">Opción 1</li>
    <li role="listitem" aria-level="1">Opción 2</li>
    <li>
        <span>Opción 3</span>
        <ul role="list">
            <li role="listitem" aria-level="2">Opción 3.1</li>
            <li role="listitem" aria-level="2">Opción 3.2</li>
        </ul>
    </li>
</ul>
```

1.3.6.3.17 Atributo aria-live

El atributo ARIA-LIVE permite identificar aquellas zonas del documento que pueden ser actualizadas de forma dinámica o automática. Esto, hace posible que se les notifique a las herramientas de asistencia situaciones en las que se inyectan contenidos dinámicamente a un contenedor actualizable. Las herramientas de asistencia leen automáticamente el contenido de este contenedor (denominado región activa o "Live Region") y evitan tener que centrarse en dónde se producen los cambios.

Recordemos que, si el valor del atributo ARIA-LIVE es POLITE, el contenido podrá ser actualizado una vez haya acabado las tareas que esté haciendo en ese momento el usuario, pero, si el valor del atributo es ASSERTIVE, el contenido podrá ser actualizado, aunque el usuario no haya terminado las tareas en ese momento.

Este atributo suele combinarse con el atributo ARIA-ATOMIC para indicar si la actualización implica a toda la región o sólo a partes concretas. También,

suele combinarse con el atributo ARIA-RELEVANT, el cual, indica qué tipo de actualización debe realizarse o se espera.

1.3.6.3.18 Atributo aria-orientation

El atributo ARIA-ORIENTATION permite indicar, a las herramientas de asistencia, que si el elemento y su orientación es vertical u horizontal. Sus posibles valores son HORIZONTAL y VERTICAL.

Ejemplo:

```
<label for="temperature" id="tempLabel" class="label">Temperatura</label>
<div id="temperature"
     role="slider"
     aria-labelledby="tempLabel"
     aria-orientation="vertical"
     aria-valuenow="25"
     aria-valuetext="25"
     aria-valuemin="-20"
     aria-valuemax="60">
</div>
```

1.3.6.3.19 Atributo aria-modal

El atributo ARIA-MODAL permite indicar, a las herramientas de asistencia, que la acción de pulsar en el elemento conllevará la apertura de una estructura que interrumpirá el flujo de trabajo actual. Es decir, conllevará la apertura de un diálogo modal.

En general, este atributo suele estar asociado al rol DIALOG o ALERTDIALOG, lo que significa, entre otras cosas, que se debe colocar el foco en el primer elemento del diálogo en el momento de ser visualizado, a menos, eso sí, de que el foco se haya establecido explícitamente en otro lugar.

Ejemplo:

```
<h3>Newsletter</h3>
<div role="dialog" aria-modal="true" id="newsletter">
    <form action="post">
        <label for="estado">Email:</label>
        <input type="email" id="email" />

        <input type="submit" value="Enviar" />
    </form>
</div>
```

1.3.6.3.20 Atributo aria-multiselectable

El atributo ARIA-MULTISELECTABLE permite indicar, a las herramientas de asistencia, que el usuario puede seleccionar más de un elemento de las opciones ofrecidas a continuación. Como norma, este atributo va asociado con el atributo ARIA-SELECTED.

En general, se suele establecer en desplegables, listas y árboles, pero se puede utilizar en cualquier elemento de interacción que permita múltiples valores.

Ejemplo:

```
<label for="estado">Aficiones/Hobbies:</label>
<select id="estado" aria-multiselectable="true">
    <option value="0">Fútbol</option>
    <option value="1" selected aria-selected="1,2">Música</option>
    <option value="2" selected aria-selected="1,2">Cine</option>
    <option value="3">Teatro</option>
</select>
```

1.3.6.3.21 Atributo aria-multiline

El atributo ARIA-MULTILINE permite indicar, a las herramientas de asistencia, que el control de entrada permite varias líneas. Esto es aplicable tanto para elementos TEXTAREA, como para elementos con el atributo CONTENTEDITABLE establecido a TRUE.

Ejemplo:

```
<div contenteditable="true" aria-multiline="true"></div>
```

1.3.6.3.22 Atributo aria-owns

El atributo ARIA-OWNS permite indicar, a las herramientas de asistencia, que el elemento está vinculado o asociado con otro elemento que se encuentra fuera de la estructura actual o separado en otra zona del documento.

Esto es útil, por ejemplo, cuando se están definiendo menús de navegación que poseen varios niveles.

Ejemplo:

```
<nav role="navigation">
    <ul role="menu">
        <li role="menuitem" aria-owns="submenu-new">Nuevo</li>
        <li role="menuitem">Abrir</li>
        <li role="menuitem">Guardar</li>
        <li role="menuitem">Guardar como ...</li>
```