2024/25

U.D.6. DISEÑO DE WEBS ACCESIBLES

ÍNDICE

1. Concepto de accesibilidad web

La **accesibilidad web** consiste en desarrollar **aplicaciones web** que puedan ser **utilizadas** por el mayor número de **usuarios con necesidades específicas**. Dichas necesidades pueden ser debidas a limitaciones **derivadas del entorno** o **derivadas del propio usuario, como son los problemas visuales, auditivos, motrices y neurológicos** (dislexia, trastornos de atención, falta de memoria …).

Por tanto, el acceso al portal web debe facilitarse para todos los usuarios potenciales, más allá de las limitaciones técnicas de cada usuario (software, hardware, etc.) o de las limitaciones individuales de cada uno (discapacidades, dominio de un determinado idioma, etc.). De esta forma, un sitio web accesible debe tener en cuenta la **gran diversidad de potenciales usuarios** que puede llegar a tener.

1.1 Barreras derivadas del entorno

Las **barreras derivadas del entorno** más habituales son las siguientes:

- **Navegadores web antiguos**. Para superar esta barrera debemos **proporcionar alternativas** cuando se utilizan elementos que no tienen soporte en tecnologías antiguas. Si usamos Javascript para mostrar un menú, este debe funcionar igualmente, aunque la tecnología no esté disponible.

- **Navegadores de texto**. Para superar esta barrera debemos **incluir un equivalente textual** para todos los elementos no textuales (imágenes, vídeos o sonidos).

- **Conexiones lentas**. Para superar esta barrera debemos **minimizar el tiempo de carga** de los elementos de la web.

- **Pantallas pequeñas o muy grandes**. Para superar esta barrera debemos tener un **diseño web responsive**.

- **Monitores monocromos**. Para superar esta barrera debemos **evitar funcionalidades que se interpreten por su color**, por ejemplo.

1.2 Barreras por tipo de perfil

Las barreras por tipo de perfil son aquellas que dependen propiamente del usuario como persona. Las principales son las que se detallan a continuación.

1.2.1 Ceguera y baja visión

La ceguera se define como la **limitación total de la función visual**, mientras que la **baja visión** es una limitación parcial de la función visual (no es total).

Las principales **barreras** que se encuentran las personas con ceguera y baja visión son las siguientes:

a) Las **imágenes o elementos no textuales sin texto alternativo** no pueden ser leídas por los lectores de pantalla.

b) Las **imágenes que incluyen gráficos que representan datos** o **textos insertados mediante imágenes** tampoco son leídos por los lectores de pantalla.

c) **Elementos multimedia** sin descripción textual.

d) **Tablas en las que el contenido es incomprensible** cuando se lee de forma secuencial.

e) **Formularios y tablas de datos complejos y difíciles de interpretar** correctamente. Cuando se crean formularios, es especialmente necesario cuidar ciertos aspectos de accesibilidad para poder garantizar una correcta interpretación de los contenidos a los usuarios de lectores de pantalla.

f) Falta de independencia de dispositivo o **imposibilidad de acceder al contenido, o de operar con la aplicación web, desde el teclado.** La **web** debe ser **funcional cuando no se utilice ratón**.

g) **Formatos no accesibles de documentos** que pueden dar problemas a los lectores de pantalla si no cumplen las normas de accesibilidad (por ejemplo, un documento pdf que no cumple las normas).

h) **Utilización inadecuada de elementos estructurales** en las páginas **o falta de estructuración** en sus contenidos: ausencia de encabezados de sección, definiciones de listas, agrupaciones de controles, etc.

> **¿Por qué hay que poner especial atención a los encabezados?**
>
> Su función es describir brevemente el tema de la sección que introducen. Estas son situaciones claras donde deberían existir encabezados de sección (h1 a h6).
>
> Gracias al uso correcto de este tipo de elementos estructurales, las ayudas técnicas pueden **facilitar la navegación** a aquellos usuarios que acceden de forma lineal a los contenidos de una página o documento. Este es el caso de usuarios ciegos, para los **cuales la posibilidad de navegar por los encabezados es enormemente útil**.

i) Sitios con **información basada en el color**.

j) **Presencia de captchas en los que no se aporta solución accesible**. En el caso de las imágenes de texto visualmente distorsionadas, que se usan como mecanismos de control destinados a distinguir a un humano de una máquina o programa de ordenador, se deben proporcionar distintos métodos alternativos para acceder a la información adaptados a diferentes capacidades sensoriales.

k) Sitios con **pobre contraste entre textos, fondos e imágenes** (para baja visión).

l) **Tamaño de letra con medidas absolutas** que no permiten su cambio (para baja visión).

m) **Maquetación desajustada** al modificar los tamaños de la fuente y que complican la navegabilidad.

1.2.2 Daltonismo

El daltonismo es una alteración de origen genético que afecta a la capacidad de distinguir los colores. Con frecuencia no se distinguen los verdes de los rojos y, a veces, los azules.

Las principales **barreras** que se encuentran las personas con daltonismo son la siguientes:

a) **Color para el resaltado** de los textos sin utilizar otro formato adicional como la cursiva, la negrita o el subrayado.

b) **Poco contraste** entre textos, fondos e imágenes.

1.2.3 Auditivas

La discapacidad auditiva se define como el **déficit total o parcial en la percepción del sonido**, que se evalúa por el grado de pérdida de la audición en cada oído. Se puede distinguir entre las personas hipoacúsicas, que son las que presentan una deficiencia parcial y que pueden mejorar a través de audífonos, y por otro lado, las personas sordas, las cuales presentan una deficiencia total.

Las principales **barreras** que se destaca en este caso son:

a) El **incipiente uso de elementos audiovisuales** en la web (vídeos, animaciones, sonidos, etc.) sin **una alternativa de texto como los subtítulos o las transcripciones.**

b) **La obligatoriedad del uso de micrófono** sin posibilidad de desactivación.

1.2.4 Motrices

Una persona con una discapacidad motora es aquella que sufre de una manera duradera, y frecuentemente crónica, una **afección más o menos grave del aparato locomotor** que supone una limitación de sus actividades en relación con el promedio de la población. Se debe conocer también que esta **cubre todos los trastornos que pueden causar deterioro**

parcial o total de las habilidades motoras, incluyendo la parte superior e inferior del cuerpo.

Como consecuencia de esto, en términos relativos al diseño de un sitio web, puede haber personas con **dificultades** para llevar a cabo algunas tareas informáticas como **utilizar un ratón, mover un puntero, utilizar una pantalla táctil, pulsar dos teclas al mismo tiempo o mantener pulsada una tecla**, e incluso pueden ser incapaces de utilizar un teclado y, por ello, no poder introducir datos.

Por tanto, las principales barreras que se destaca en este caso son las siguientes:

a) Elementos de interacción muy pequeños: botones, enlaces, etc.
b) **Imposibilidad de interaccionar adecuadamente con la página desde el teclado u otros dispositivos de entrada**. Para facilitar la interacción con las páginas web se recomienda usar dispositivos de entrada diseñados especialmente para ser utilizados con el ordenador.
c) **Falta de independencia de dispositivo**; la web debe ser funcional cuando no se utilice ratón.
d) Enlaces **gráficos y otros elementos accionables que no están etiquetados correctamente y no son accesibles a los reconocedores de voz.**

1.2.5 Neurológicas o cognitivas (dislexia, trastornos de atención, falta de memoria...)

Podemos decir que las habilidades cognitivas son aquellas que nos facilitan la adquisición de conocimiento, su retención y su recuperación.

Las **habilidades cognitivas** son las siguientes:

- **Observación.** Todo aquello relacionado con la atención, concentración, búsqueda e identificación de datos, elementos u objetos.
- **Análisis**. Todo aquello relacionado con las habilidades para hacer comparaciones y distinciones, con la finalidad de distinguir y destacar los elementos básicos de información.

- **Ordenación**. Todo aquello relacionado con la agrupación, reunión y serialización que permite, a partir de un atributo determinado, disponer un conjunto de datos de forma sistemática.
- **Clasificación**. Todo aquello relacionado con la categorización, esquematización, jerarquización de un conjunto de datos.
- **Representación**. Todo aquello relacionado con la modelización, reproducción o simulación como medio de recreación de nuevos hechos o situaciones a partir de los existentes.
- **Memorización**. Todo aquello relacionado con la retención, conservación, evocación, almacenamiento y recuperación de datos.
- **Interpretación**. Todo aquello relacionado con la argumentación, la deducción y el razonamiento que nos permite aportar un significado personal a la información.
- **Evaluación**. Todo aquello relacionado con la estimación, la crítica y el juicio que nos permite realizar valoraciones sobre los datos obtenidos.

Por tanto, las personas con discapacidad cognitiva presentan **dificultades** en el desarrollo de la **inteligencia verbal y matemática**, pueden tener un **menor rendimiento en la lectura**, en la **precisión**, en la **comprensión** o en la **velocidad**, lo que ocasiona trastornos del aprendizaje.

Las principales **barreras** que se destaca en este caso son las siguientes:

a) **Tamaño de letra fijo** que no se puede cambiar.
b) **Elementos sonoros o visuales que no se pueden desactivar**.
c) **Falta de estructuración y organización** del contenido que impide entenderlo correctamente.
d) **Lenguaje muy enrevesado** y frases muy complejas.
e) **Destellos o parpadeos** frecuentes que pueden provocar **ataques de epilepsia**.

1.2.5.1 La epilepsia fotosensitiva

La **epilepsia fotosensitiva** es un problema causado por una **respuesta anormal del cerebro a las luces intermitentes** (tipo flash). Se debe a que el mecanismo cerebral que controla la reacción a la información visual “es defectuosa o está ausente”.

Además de las luces parpadeantes o relampagueantes rápidas, **especialmente si son rojas**, los ataques pueden ser causados, a veces, por ciertas **formas y patrones geométricos**. La frecuencia del parpadeo de luz que provoca estos ataques varía de persona a persona. Generalmente se da en frecuencias que oscilan entre los **5 y los 30 parpadeos por segundo** (hertz), esto es, varía de persona a persona.

Ejemplos de víctimas de epilepsia televisiva:

1) "Más de 700 niños tienen que ser hospitalizados en Japón tras ver una serie de dibujos animados." Ver noticia en el Diario Información

2) Este episodio de Pokémon causó ataques de epilepsia: https://youtu.be/VbaOKaEjUsg

Puedes ver más desafíos y soluciones de diseño web accesible en la siguiente url:
Accesibilidad Web: La persona discapacitada y la web (ua.es)

2. Tecnología asistencial

La **tecnología asistencial** es el conjunto de equipos, dispositivos, instrumentos o programas empleados con la finalidad de mejorar la calidad de vida de aquellas personas que tienen algún tipo de discapacidad incrementando así su **autonomía**.

2.1 Dispositivos de entrada

Los dispositivos de entrada considerados como tecnología asistencial son aquellos que **permiten realizar las mismas funciones** que se realizarían con un teclado o con un ratón convencional. Los más conocidos son:

1) **Teclado virtual.**

   Este dispositivo facilita el trabajo a aquellas personas que no pueden usar un teclado con normalidad. Algunos teclados virtuales incorporan un sistema predictivo de palabras con lo cual el esfuerzo para escribir es menor.

2) **Teclado alternativo.**

   Son teclados **adaptados** a las distintas necesidades. Las adaptaciones realizadas pueden ser:

   - Un **aumento del tamaño físico de las teclas o del espacio** entre las teclas para facilitar su uso por las personas con discapacidad motriz.
   - Un **aumento del tamaño de las letras** escritas sobre la tecla para facilitar la visión a aquellas personas con discapacidad visual.
   - Un **teclado coloreado o con imágenes** para facilitar el aprendizaje en las personas con discapacidad cognitiva. Estos teclados pueden tener una retroalimentación auditiva, indicando mediante el sonido cuál es la tecla pulsada.

3) **Línea y teclado Braille**.

   La línea braille es un periférico que dispone de unas celdas que permiten representar caracteres braille y ser leídos por personas ciegas mediante los dedos.

   Una **línea y teclado braille** es un dispositivo de **entrada y salida** que combina en un solo dispositivo una línea de braille y un teclado braille.

Puedes ver cómo funciona en el siguiente vídeo:
https://www.youtube.com/watch?v=G69n-1mgcTk

4) **Software de reconocimiento de voz**.

Empleado para poder introducir datos o ejecutar comandos en el ordenador a aquellas personas que no pueden hacer uso del teclado ni del ratón. Para ello el ordenador emplea sus **funciones de audio**. Los sistemas operativos y algunos navegadores ya dan soporte al reconocimiento de voz sin necesidad de instalar un software adicional.

5) **Licornios o Apuntadores.**

Estos dispositivos están pensados para aquellas personas que no tienen movilidad en las extremidades pero sí en la cabeza. Es un **casco** que lleva incorporada una varilla larga acoplada en la frente o en la barbilla y que es empleada para realizar pulsaciones en el teclado. Estos apuntadores deben complementarse con una función conocida como "**sticky keys**" que permite simular la pulsación simultánea de varias teclas pero pulsándolas de una en una.

6) **Trackball gigante.**

Es un ratón que no necesita desplazarse. Tiene una bola de gran tamaño situada por encima del ratón que se mueve en cualquier dirección y unos botones de gran tamaño facilitando, de esta forma, su uso por las personas con discapacidad motriz. Algunos disponen de un botón adicional que permite ampliar las imágenes con lo que se convierte en una herramienta útil también para las personas con discapacidad visual.

Además del trackball existen otros ratones especiales como los **ratones de cabeza**, ratones **de pie o apuntadores de boca**, ratón **de mirada**, ratón **de palanca**, etc.

7) **Webcams para seguimiento de ojos o cara.**

Estas webcams están preparadas para transformar el movimiento de los ojos o la cara en movimientos del puntero del ratón en la pantalla simulando las pulsaciones del

ratón con un parpadeo o con un gesto concreto de la cara. Estos dispositivos son adecuados para aquellas personas que tienen una discapacidad motriz severa de las extremidades y además tienen dificultad en el habla como para poder emplear el software de reconocimiento de voz.

2.2 Dispositivos de salida

Los dispositivos de salida más conocidos empleados como tecnología asistencial son:

- **Lectores de pantalla.**

  Son aplicaciones software que **leen el texto de la pantalla en voz alta** mediante un sintetizador de voz. También **pueden enviar el texto a una línea braille** para que el usuario lo lea con los dedos.

  Son programas especialmente útiles para las **personas con ceguera o con una discapacidad visual muy grande** ya que permiten hacer una lectura de lo que se muestra por pantalla mediante un sintetizador de voz.

  La aplicación software más famosa es **JAWS** y se maneja con las teclas del teclado. Estos son algunos de sus comandos:

  - INSERT+FLECHA ABAJO: leer todo.
  - ESC: detener la lectura.
  - FLECHA IZQUIERDA: carácter anterior.
  - TAB: se mueve al siguiente enlace.
  - SHIFT+TAB: se mueve al enlace anterior.
  - V: se mueve al siguiente enlace visitado, etc.

  > Puedes ver cómo funcionan los lectores de pantalla JAWS y NVDA en Lectores de pantalla JAWS y NVDA.pdf (ua.es)

- **Ampliadores de pantalla.**

  Son programas especialmente útiles en las personas con baja visión ya que permiten ampliar el texto y las imágenes mostrados en el monitor.

- **Líneas Braille** (ya mencionadas).

  Están compuestas por una serie de celdas con 6 u 8 puntos cada una que muestran de forma táctil la misma información que leería un lector de pantalla. Son útiles para las personas ciegas y, sobre todo, para las **sordo-ciegas** que no se pueden beneficiar de los lectores de pantalla ni de los ampliadores de pantalla.

- **Navegadores para ciegos.**

  Son capaces de leer las páginas web en voz alta y los usuarios podrán navegar a través de su voz haciendo uso de una serie de comandos especiales.

En el mundo de la web **todos somos responsables de la accesibilidad**: personas que se dedican al diseño web, empresas que fabrican navegadores y lectores de pantalla o que distribuyen software y crean las herramientas empleadas por los que diseñan sitios web e, incluso, las personas con discapacidades que usan la web como responsables del empleo de la tecnología asistencial.

Puedes ver más tecnología asistencial en:
http://accesibilidadweb.dlsi.ua.es/?menu=hardware

3. El consorcio World Wide Web (W3C)

El consorcio **World Wide Web** (W3C) es una comunidad internacional donde las organizaciones miembro se encargan del **desarrollo de estándares** que aseguran el crecimiento y el acceso a la web.

Fue creada en 1994 con un conjunto de objetivos que permitieran desarrollar tecnologías interoperables.

Dentro de este marco se hace necesario desarrollar estrategias, directrices y recursos para garantizar el acceso por igual a la web, es así cómo aparece la ***Web Accessibility Initiative*** o Iniciativa para la accesibilidad a la web (**WAI**). Esta iniciativa desarrolló las **Directrices de accesibilidad** para el contenido web 2.0, más conocido como **WCAG 2.0**, donde se recogen las pautas y las técnicas que permitan **ofrecer soluciones accesibles para el software y contenido web.** Este conjunto de pautas fue aprobado bajo el estándar internacional ISO/IEC 40500:2012.

- Versión original: *https://www.w3.org/TR/WCAG20/*
- Versión en castellano: *http://sidar.org/traducciones/wcag20/es/*

En junio de 2018 fue sustituido por la versión **WCAG 2.1**; y en octubre de 2023 la versión WCAG 2.1 fue sustituida por la versión **WCAG 2.2**.

3.1 WCAG 1.0

Las guías WCAG proporcionan unas pautas de diseño que ayudan a que un sitio web sea accesible. En la primera versión de las guías WCAG (WCAG 1.0) se identificaron **14 pautas de diseño**.

1) Proporcionar alternativas equivalentes para contenido visual y auditivo.
2) No basarse solo en el color.
3) Usar marcadores y hojas de estilo y hacerlo de forma correcta.
4) Especificar el lenguaje utilizado.
5) Crear tablas que se transformen correctamente.
6) Asegurarse de que las páginas que usen nuevas tecnologías se transformen correctamente.
7) Asegurar al usuario el control de los contenidos que cambian con el tiempo.
8) Asegurar la accesibilidad directa de las interfaces incrustadas.

9) Diseñar para la independencia del dispositivo.
10) Usar soluciones provisionales.
11) Usar las tecnologías y pautas de la W3C.
12) Proporcionar información de contexto y orientación.
13) Proporcionar mecanismos claros de navegación.
14) Asegurar documentos claros y simples.

3.2 WCAG 2.0, 2.1 y 2.2

En esta última revisión, las catorce pautas se redujeron a cuatro.

Los cuatros principios que regulan este funcionamiento son que el diseño debe ser **perceptible, operable, comprensible y robusto**.

1. **Principio 1. Diseño perceptible.** La información y los componentes de la interfaz de usuario deben ser mostrados a los usuarios de manera que pueda ser entendida. Para ello se establecen cuatro directrices:
   a) **Texto alternativo**. Proporcionar texto alternativo para el contenido que no sea textual (caracteres grandes, lenguaje braille, lenguaje oral, símbolos o lenguaje más simple).
   b) **Alternativas parar el contenido multimedia** o contenido multimedia dependiente del tiempo.
   c) **Adaptable**. Crear contenido que pueda ser presentado de diferentes formas sin perder información o estructura.
   d) **Distinguible**. Facilitar a los usuarios la posibilidad de ver y escuchar el contenido incluyendo la distinción entre lo más y menos importante.

2. **Principio 2. Diseño operable.** Los componentes de la interfaz de usuario y la navegación deben ser manejables. Para ello se establecen **cuatro directrices**:
   a) **Teclado accesible**. Poder controlar todas las funciones desde el teclado.
   b) **Tiempo suficiente**. Proporciona tiempo suficiente a los usuarios para leer y utilizar el contenido.
   c) **Ataques epilépticos**. No diseñar contenido que pueda causar ataques epilépticos, espasmos o convulsiones.

d) **Navegación**. Proporciona formas para ayudar a los usuarios a navegar, a buscar contenido y a determinar dónde están estos.

3. **Principio 3. Diseño comprensible**. La información y las operaciones de los usuarios deben ser comprensibles. Para ello se establecen **tres directrices**:
   a) **Legible**. Hacer contenido de texto legible y comprensible.
   b) **Previsible**. Presentar una apariencia y una forma de utilizar las páginas web previsibles.
   c) **Asistencia** a la entrada de datos. Ayudar a los usuarios a evitar y corregir errores.

4. **Principio 4. Diseño robusto.** El contenido ha de ser robusto para que pueda ser interpretado por una gran variedad de agentes de usuario, incluyendo las tecnologías de asistencia. Para ello se establece **una directriz**:
   a) **Compatible**. Maximizar la compatibilidad con las aplicaciones de usuario actuales y futuros, incluyendo las tecnologías de asistencia.

Las pautas de las WCAG 2.0 se pueden consultar en w3.org/TR/WCAG20/, las WCAG 2.1 en w3.org/TR/WCAG21/ y las WCAG 2.2 en w3.org/TR/WCAG22/. También puedes ver un resumen en español de las distintas versiones en la página web sobre accesibilidad de la Universidad de Alicante.
Se prevé que la próxima versión sea la WCAG 3.0.

IMPORTANTE:

- Si se cumple con las **WCAG 2.2 se cumple también con las WCAG 2.1**. La WCAG 2.2 es una nueva ampliación de 9 criterios, donde además se elimina el criterio 4.1.1.

- Cuando se publique en el Diario Oficial de la Unión Europea la nueva versión de la **EN 301 549** que recoja los nuevos requisitos de las WCAG 2.2, entonces serán de **obligado cumplimiento** en España y la Unión Europea.
  Más información en: https://olgacarreras.blogspot.com/2020/08/wcag-22-novedades-del-ultimo-borrador.html

3.2.1 Niveles de prioridad

La WCAG ofrece varios niveles de prioridades a la hora de aplicar las pautas de accesibilidad. De esta forma podemos saber cuáles son las acciones más importantes que debemos realizar. Son **tres niveles** de **prioridades**:

- **Prioridad 1**. Nivel mínimo exigible y requisito esencial que se debe satisfacer. De otra forma, uno o más grupos de usuarios encontrarán imposible acceder a la información.
- **Prioridad 2**. Satisfacer este punto de verificación eliminará importantes barreras de acceso a la información Web.
- **Prioridad 3**. Satisfacer este punto de verificación mejorará la accesibilidad a la información web a usuarios que tengan dificultades para utilizar la interfaz.

3.2.2 Niveles de conformidad

En la WCAG 2.0, 2.1 y 2.2 se recogen un conjunto de **criterios de conformidad**, redactados en forma de enunciados verificables sobre el contenido web, y que son utilizados para verificar la adecuación a la accesibilidad de un sitio web.

Se distinguen **tres niveles de conformidad** que **definen el grado de accesibilidad.** Por tanto, para que una página web sea conforme con las **WCAG 2.2** deben satisfacerse todos los **requisitos o niveles de conformidad** siguientes:

- **Nivel A** (prioridad 1): para lograr conformidad con el nivel A (el mínimo), la página web satisface todos los criterios de conformidad del nivel A, o proporciona una versión alternativa conforme. Se deben satisfacer **31 criterios**.
- **Nivel AA** (prioridades 1 y 2): para lograr conformidad con el nivel AA, la página web satisface todos los criterios de conformidad de los niveles A y AA, o se proporciona una versión alternativa conforme al nivel AA. Se deben satisfacer **24 criterios** (además de los 31 del nivel A).
- **Nivel AAA**: para lograr conformidad con el nivel AAA, la página web satisface todos los criterios de conformidad de los niveles A, AA y AAA, o proporciona una versión alternativa conforme al nivel AAA. Se deben satisfacer **31 criterios** (además de los 31 del nivel A y los 24 del nivel AA).

En la siguiente guía se puede ver en de forma simplificada los niveles de conformidad de la WCAG 2.2: https://guia-wcag.com/es/.

3.3 WCAG 3.0

El 21 de enero de 2021 se publicó **el primer borrador de las** W3C Accessibility Guidelines (WCAG) 3.0. Las WCAG 3 nos traen muchas novedades, comenzando por el nombre: "W3C Accessibility Guidelines" en vez de "Web Content Accessibility Guidelines". Este cambio permite conservar el acrónimo WCAG, pero ahora refleja un alcance más amplio.

La elaboración y publicación de las WCAG 3 es un **proceso que llevará años completar**. El nuevo estándar introduce cambios significativos en diversos aspectos: filosofía, alcance, estructura, contenido o modelo de conformidad. Aunque las WCAG 3.0 están diseñadas para ser más fáciles de entender y más fáciles de utilizar, será **complejo familiarizarse y adaptarse a auditar con el nuevo estándar**

Podéis encontrar más información en los siguientes enlaces:

- **W3C Accessibility Guidelines (WCAG) 3.0**
- WCAG 3 Introduction, W3C
- Explainer for W3C Accessibility Guidelines (WCAG) 3.0, documento al que se movió gran parte de la introducción sobre las WCAG 3.0 de la primera versión del borrador

3.4 Aspectos a tener en cuenta

A la hora de aplicar las técnicas para satisfacer los requisitos definidos en las WCAG, debemos tener en cuenta los siguientes aspectos.

1) ***Páginas completas.*** La conformidad se aplica a páginas web completas, y no se puede alcanzar si se excluye una parte de la página.

2) ***Procesos completos***. Cuando una página web es parte de una serie de páginas web que presentan un proceso, todas las páginas en ese proceso deben ser conformes con el nivel especificado o uno superior.

3) ***Uso de tecnologías exclusivamente según métodos que sean compatibles con la accesibilidad***. Para satisfacer los criterios de conformidad solo se depende de aquellos usos de las tecnologías que sean compatibles con la accesibilidad.

4) ***Sin interferencia***. Si las tecnologías se usan de una forma que no es compatible con la accesibilidad, o están usadas de una forma que no cumplen los requisitos de conformidad, no deben impedir a los usuarios acceder al contenido del resto de la página.

3.5 Técnicas para satisfacer los requisitos definidos en las WCAG

Las técnicas para satisfacer los requisitos definidos en las WCAG también están recogidas en la web w3.org. Hay más de 4000 técnicas para satisfacer todos los requisitos definidos en las WCAG. A continuación, se mencionan las más destacadas.

3.5.1 Técnicas fundamentales

- Incluir **alternativas de texto** al contenido **multimedia** (subtítulos).
- **Permitir pausar** el contenido multimedia.
- Incluir **alternativas legibles** cuando la lectura dependa del contraste de colores.
- **Ordenación** coherente del contenido.
- Incluir **glosarios**, **mapa del sitio**, **tabla de contenidos**.
- Incluir **títulos descriptivos**.
- Añadir **enlaces para ir al principio** de la página.
- Identificar la localización del usuario dentro de la web (**breadcrumbs**).
- **Evitar parpadeos**.
- Alinear los **textos de manera similar**.
- Ofrecer **feedback** de confirmación o negación al realizar una operación, etc.
- Emplear un **lenguaje claro y sencillo**.

3.5.2 Técnicas HTML

- Emplear un código **HTML semánticamente correcto**.
- Utilizar **etiquetas** estructurales o elementos **semánticos** (*<header>*, *<nav>*, *<section>*, *<article>*, *<footer>*, *<aside>*…)
- Utilizar elemento *title* para dar un título coherente a la página.
- Incluir la etiqueta *meta description*.
- Incluir el **botón de** *submit* en los formularios.
- **Definir el idioma** correspondiente con el **atributo** *lang*.
- Incluir el atributo *alt* **en las imágenes**.
- Utilizar la etiqueta **caption en las tablas**. Del mismo modo, también deben usarse las etiquetas *thead*, *tbody* y *tfoot* para identificar la cabecera, el cuerpo y el pie de las tablas HTML.
- **Maquetación sin tablas**.
- Utilizar los **encabezados** *h1*-*h6*.
- Crear un **orden de tabulación coherente** en formularios y enlaces.
- Dar **nombres significativos a los enlaces** para que puedan ser leídos correctamente por los lectores de pantalla.
- Para toda aquella **información relacionada**, **utilizar listas en vez de párrafos separados**, ya que los lectores de pantalla no serán capaces de identificar que se trata de elementos relacionados.
- Si es necesario, utilizar el atributo *tabindex* para cambiar el orden mediante el cual los elementos mostrados en pantalla se seleccionan cuando se pulsa el tabulador. El atributo *tabindex* acepta un **número** como valor que **se corresponde con el orden de selección**. Si se agrega *tabindex="0"* a un elemento, será posible seleccionarlo siguiendo el flujo de

```
<div tabindex="0">
  ...
</div>
```

selección habitual. Si se agrega `tabindex="-1"` a un elemento, ya no será posible seleccionarlo o saltar a él cuando se pulse el tabulador

3.5.3 Técnicas CSS

- Facilitar **mecanismos para que se pueda modificar la hoja de estilos** CSS: colores, fuentes, etc.
- **Separar la estructura de los estilos** y **no utilizar estilos en línea**.
- Utilizar tamaño de letra con **medidas relativas**.
- Incluir el foco en los elementos mediante la pseudo-clase **`:focus`.**
- **Utilizar CSS para presentar el texto, controlar el espaciado**, etc.
- Elegir un **tamaño de letra grande con un buen contraste** con el fondo y una **adecuada combinación de colores.**
- Crear **elementos de interacción lo suficientemente grandes** facilitaremos a las personas con alguna discapacidad motriz el poder interactuar con una página.

3.5.4 Javascript

- **Aumentar los tiempos de acción**.
- **Soporte** para utilizar tanto **ratón como teclado**.
- **Validación de datos con alertas**.
- **Usar el DOM para manipular la página**.
- Utilizar **scripts** para **modificar el aspecto de la página**, como el **fondo**.
- Utilizar **scripts** para hacer **scroll por la página y que se pueda controlar**.

3.5.5 Técnicas para servidor

- Aplicar **redirecciones en lado del servidor** (.htaccess) y no en el cliente.

3.5.6 Técnicas WAI-ARIA

Las técnicas **WAI-ARIA (Web Accessibility Initiative – Accessible Rich Internet Applications)** proporcionan semántica, de tal forma que se pueden transmitir comportamientos de la interfaz de usuario e información estructural a las tecnologías asistenciales (por ejemplo, lectores de pantalla). La especificación de ARIA establece componentes que definen **roles (*ARIA Role*), estados y propiedades (*Atributos ARIA*) de los elementos de la interfaz de usuario.**

3.5.6.1 El atributo role (ARIA Role)

El atributo ***role*** permite **asignar diferentes roles a cada uno de los elementos** de la página. Los roles permiten identificar en qué consisten los diferentes elementos de la página. Sin embargo, no todos los elementos necesitan tener asignado un rol, sino **solo aquellos más relevantes**.

Los lectores de pantalla son capaces de inferir el rol adecuado de cada elemento en la mayor parte de los casos en base al código HTML existente.

> **No será necesario que se agreguen roles a las etiquetas HTML que ya identifican semánticamente su contenido**, como por ejemplo las etiquetas *nav*, *form*, *button*, *table*, *ul* o *li*.

Por ejemplo, si por algún motivo nos vemos obligados a usar una etiqueta *div* en lugar de una etiqueta *nav*, tendríamos que usar el rol *navigation* en dicha etiqueta:

```
<div role="navigation">
  <ul>
    <li><a href="/">Inicio</a></li>
    <li><a href="/noticias">Noticias</a></li>
    <li><a href="/contacta">Contacta</a></li>
  </ul>
</div>
```

> Por tanto, el **atributo *role* se usa para dar significado a aquellos elementos de la interfaz que carecen del mismo**.

A continuación, se nombran algunos de los roles más utilizados. Se puede encontrar la **lista completa**, así como las recomendaciones, **en la página web de la W3C**: https://www.w3.org/TR/html-aria/

Por otro lado, se pueden ver **ejemplos** de uso en https://www.w3.org/TR/aria-in-html/

⇒ `complementary`. Sirve para **identificar una sección de la página que se relaciona con el contenido principal** de la misma, pero que también podría tener sentido de forma independiente. Un ejemplo de ello sería una barra lateral con enlaces o información relacionada con el artículo principal. **Siempre que sea posible, debe usarse la etiqueta semántica `aside` en estos elementos.**

⇒ `list`. Sirve para **identificar una lista de elementos**. Suele usarse junto con el rol `listitem`, usado para identificar los elementos de la lista.

⇒ `listitem`. Se utiliza para **identificar los elementos que se encuentran en el interior de los elementos que tienen el rol list**, usados a modo de contenedor.

⇒ `main`. Se usa para identificar el **contenido principal de un documento**. Se trata del contenido central de la página, sin el cual no tendría sentido su existencia.

⇒ `navigation`. Se usa para identificar aquellos **grupos de enlaces** que se utilizan para navegar por el sitio web o por los contenidos de la propia página.

⇒ `region`. Se utiliza para **identificar áreas del documento** que los autores han identificado como **relevantes**, proporcionando un mecanismo de navegación sencillo cuando ningún otro rol se adapta.

⇒ `tab`. Sirve para **indicar** aquellos **elementos interactivos dentro de una lista de pestañas** que mostrarán su respectivo contenido al hacer clic en ellos.

⇒ `tabpanel`. Se usa para **identificar los paneles de contenido** con los que se corresponde cada elemento con el rol `tab`.

⇒ ***alert***. Puede usarse para **proporcionar ciertos mensajes al usuario**. Por ejemplo, se usará cuando algún elemento se actualice en pantalla. El lector de pantalla leerá el mensaje automáticamente y lo cerrará.

⇒ ***alertdialog***. Funciona **exactamente igual que el rol *alert***. La única diferencia reside en que es el usuario el que deberá cerrar el mensaje de alerta.

⇒ ***application***. Sirve para indicar que **el elemento que lo incluye y todos sus descendientes deben ser tratados igual que una aplicación** de escritorio, saltándose las técnicas de interpretación HTML tradicionales.

⇒ ***article***. Sirve para **indicar la sección de la página que podría conformar la página por sí misma** dentro del documento. También se suelen incluir en su interior contenidos relacionados como información acerca del autor, comentarios o noticias.

⇒ ***heading***. Se usa para **definir la cabecera** de una página o de una sección, agregando más información estructural a la página.

⇒ ***banner***. El rol banner se usa para **representar contenido genérico** dentro de una página, que **suele estar situado en la cabecera** de la misma. En el interior del elemento con este rol podría estar el nombre del sitio web, el logotipo del mismo o ciertos elementos destacados.

⇒ ***form***. Se utiliza en aquellos elementos que engloban a otros que proporcionan la misma funcionalidad que un formulario.

⇒ ***button***. Se usa, tal y como su nombre indica, para **representar aquellos elementos en los que el usuario puede hacer clic**. En un lector de pantalla, estos elementos se representarán como si fuesen botones.

⇒ ***textbox***. Se usa para identificar aquellos **elementos en los que es posible introducir texto**.

⇒ ***checkbox***. Se usa para **indicar elementos interactivos que es posible seleccionar** o marcar. Estos elementos deben contener también el **atributo *aria-checked*** para identificar así su estado.

⇒ ***table***. Se utiliza para identificar los **elementos que tienen estructura tabular no interactiva**, con sus contenidos organizados en filas y columnas, sin que necesariamente tenga que tratarse de un elemento de tipo table.

⇒ ***cell***. Sirve para indicar aquellos **elementos que forman parte de un conjunto de elementos que se encuentran en un mismo contenedor**, sin que este contenga información acerca de las cabeceras de las filas o de las columnas del mismo.

⇒ ***row***. Se usa para identificar un **conjunto de celdas organizadas en filas con estructura tabular, que no siempre han de ser tablas HTML** necesariamente. Un elemento con role row puede contener más celdas, cuadrículas o grids de celdas o cabeceras de columnas, además de una cabecera de fila.

⇒ ***rowgroup***. Se usa para indicar aquellos elementos con estructura tabular que contienen **grupos de filas** o elementos row.

⇒ ***search***. Se usa para identificar aquellos **elementos de la página que permiten buscar contenidos** en la misma, ya sean contenidos de la propia página o de otras páginas.

⇒ ***contentinfo***. Se usa para indicar aquella **información que se suele repetir al final de cada página**, como el copyright de la misma o ciertos menús con enlaces. Es habitual aplicar este rol a la etiqueta footer.

⇒ ***document***. Se suele utilizar para i**ndicar a los lectores de pantalla que deben leer el contenido que hay en su interior**. Suele usarse en widgets relativamente complejos.

⇒ ***feed***. Se usa para indicar una **lista de artículos**. Los artículos del feed se agregan o eliminan de la lista cuando el usuario hace scroll por los mismos. En muchos casos estos feeds pueden ser infinitos.

⇒ ***dialog***. Se usa para indicar aquellos **elementos que están separados de la interfaz principal** de la página, situados habitualmente sobre el resto de contenidos de la misma. Pueden ser elementos modales, pudiendo interactuar con los contenidos del

elemento dialog, o no modales, en cuyo caso todavía será posible interactuar con los contenidos que estén fuera del elemento con rol dialog.

⇒ `img`. Se usa para identificar todos aquellos **elementos que puedan ser considerados como imágenes**. Estos elementos pueden contener imágenes, emojis, texto o cualquier otro elemento visual similar a una imagen.

⇒ `figure`. Se usa para **indicar aquellos elementos visuales o esquemáticos** que no es posible englobar en otros roles, como bloques de código, conjuntos de imágenes o esquemas; todo aquello que no sea información textual en general.

⇒ `grid`. Se usa con aquellos elementos que contienen **una o más filas con celdas**. Deberá ser posible navegar por cada una de las celdas con el teclado.

⇒ `gridcell`. Se usa con los **elementos que estén en el interior de un elemento con** rol `grid`. Se usan para emular el funcionamiento de las etiquetas `td` de las tablas HTML.

⇒ `listbox`. Se usa con aquellos **elementos de entre los cuales el usuario puede seleccionar uno o más de uno**. Estos elementos son estáticos y a diferencia de las etiquetas `select`, pueden contener imágenes.

⇒ `switch`. Se usa con los **elementos cuya funcionalidad equivale a las etiquetas checkbox** de HTML, salvo porque en lugar de usar los estados checked (seleccionado) y unchecked (no seleccionado), usa los estados on (activado) y off (desactivado).

⇒ `timer`. Permite identificar los **elementos que incluyen contadores numéricos** que **cuentan el tiempo** desde un punto de inicio hasta otro final.

3.5.6.2 Atributos ARIA

Los atributos **ARIA** (***Accesible Rich Internet Applications*** o Aplicaciones Enriquecidas Accesibles de Internet) son **estados y propiedades** que proporcionan **apoyo a los roles** ARIA que existen en una página web.

Cuando se combinan con los roles, pueden suministrar información adicional de la interfaz de usuario a las tecnologías asistenciales.

Por tanto, **siempre que haya cambios** en los estados o propiedades, **las tecnologías asistenciales son notificadas** de este cambio para que puedan alertar al usuario de que se ha producido un cambio.

A continuación, vamos a ver algunos de estos **estados y propiedades**.

⇒ ***aria-label***. Este atributo se utiliza para **describir textualmente al elemento** en el que se incluye:

```
<p aria-label="Descripción del texto aquí">...</p>
```

Este atributo es **habitual en campos de texto que carecen de etiquetas**, como por ejemplo un campo de búsqueda. En ocasiones es posible que un usuario pueda inferir la utilidad de algún elemento, pero puede que un lector de pantalla lo tenga más complicado, por lo que el atributo *aria-label* resulta útil en muchos casos.

⇒ ***aria-labelledby***. Este atributo establece una **conexión entre un elemento y el elemento que lo etiqueta**. Por ejemplo, podemos tener un campo de texto *input* y una etiqueta *label*, en cuyo caso los lectores de texto serán capaces de establecer una conexión gracias al atributo *for* de la etiqueta. Sin embargo, cuando queremos establecer conexiones usando otras etiquetas HTML, debemos usar el atributo *aria-labelledby* para establecer la conexión:

```
<h2 id="titulo">Título de la película</h2>

<div aria-labelledby="titulo">
Sinópsis de la película
</div>
```

⇒ ***aria-hidden***. En ocasiones podríamos querer que los lectores de pantalla **ignoren** algunos elementos que no contienen información relevante. Para ello podemos usar el atributo *aria-hidden*, que acepta los valores **true** y **false**. Si se usa el atributo con el **valor true** en algún elemento, los lectores de pantalla lo ignorarán:

```
<div aria-hidden="true">Información poco importante</div>
```

⇒ ***aria-describedby***. Este atributo establece una **conexión entre un elemento y otro que se corresponde con su descripción**. Por tanto, establece la descripción de un elemento en el contenido de otro elemento.

```
<button aria-describedby="descripcion-boton" >Enviar formulario</button>
<div id="descripcion-boton">Haz clic en el botón para enviar el formulario</div>
```

⇒ ***aria-describedat***. Este atributo permite **especificar una URL en la que debería describirse el contenido** del elemento sobre el que se aplica el atributo:

```
<img src="/gato.png" aria-describedat="http://descripcion.com/gato" />
```

⇒ ***aria-level***. Este atributo permite establecer un **orden jerárquico** para los diferentes elementos de una estructura, que habitualmente será de **árbol**, aunque también se suele aplicar en listas de pestañas u otros elementos secuenciales. Partiendo de 1, el nivel debería aumentar a medida que aumenta la profundidad de los elementos:

```
<div id="nodo-1" aria-level="1">Nodo 1</div>
  <div id="nodo-1-1" aria-level="2">Nodo 1,1</div>
    <div id="nodo-1-1-1" aria-level="3">Nodo 1,1,1</div>
    <div id="nodo-1-1-2" aria-level="3">Nodo 1,1,2</div>
  <div id="nodo-1-2" aria-level="2">Nodo 1,2</div>
<div id="nodo-2" aria-level="1">Nodo 2</div>
  <div id="nodo-2-1" aria-level="2">Nodo 2,1</div>
  <div id="nodo-2-2" aria-level="2">Nodo 2,2</div>
```

⇒ ***aria-multiline***. Este atributo se usa para **indicar si un campo de texto acepta una o varias líneas**. Acepta los valores **true o false**, siendo false su valor por defecto:

```
<textarea id="descripcion" aria-multiline=”true” ></textarea>
```

⇒ ***aria-orientation***. Sirve para indicar si la **orientación de un elemento** es **vertical u horizontal**. Acepta los valores horizontal o vertical, siendo horizontal el valor por defecto:

```
<img src="/imagen.jpg" aria-orientation="vertical" />
```

⇒ ***aria-busy***. Sirve para **indicar si un elemento y sus descendientes están cargando datos** en el momento actual. Acepta los valores **true** o **false**, siendo false su valor por defecto:

```
<div id="usuarios" aria-busy="true">
...
</div>
```

Por ejemplo, podría ser utilizado por los autores de contenidos cuando una parte de una página web tiene contenido que se actualiza automáticamente, de manera que las tecnologías de asistencia pueden querer esperar hasta que se completen los cambios antes de informar al usuario sobre la actualización.

⇒ ***aria-flowto***. Permite **saltarse el orden lógico o visual** de los elementos e indicar el siguiente elemento al que deben saltar los lectores de pantalla. Acepta como **valor** el **id del siguiente elemento o una lista de ids**, dando al usuario la opción de escoger el siguiente elemento de entre los de la lista en este último caso:

```
<ul>
  <li id="a" aria-flowto="c">Elemento 1</li>
  <li id="b">Elemento 2</li>
  <li id="c">Elemento 3</li>
</ul>
```

⇒ ***aria-multiselectable***. Este atributo se usa para **indicar si el usuario podrá seleccionar uno o más elementos** de una lista **o de un elemento con varios descendientes**. Si su valor es false solamente se podrá seleccionar un elemento, o más de un elemento si su valor es true. Su valor por **defecto** es **false**:

```
<ul role="tablist" aria-multiselectable="true">
  <li id="tab-1">Tab1</div>
  <li id="tab-2">Tab2</div>
</ul>
```

⇒ ***aria-checked***. Este atributo se utiliza para **indicar el estado actual de un elemento que puede ser posible seleccionar**, por lo que su funcionalidad debería ser la misma que la de un elemento input de tipo checkbox. Puede tener los valores **true o false**:

```
<div id="check-option" aria-checked="true"></div>
```

⇒ ***aria-controls***. Se utiliza para indicar **aquel elemento o elementos cuyos contenidos son controlados por el elemento actual**. Acepta una referencia o id como valor.

```
<button onclick="gestionarPanel('A');" role="tab"
aria-controls="elemento-a">Selector 1</button>
<button onclick="gestionarPanel('B');" role="tab"
aria-selected="elemento-b">Selector 2</button>

<div role="tabpanel" id="elemento-a">...</div>
<div role="tabpanel" id="elemento-b">...</div>
```

⇒ ***aria-disabled***. Indica que es posible **percibir el elemento actual por pantalla, pero que está desactivado**, por lo que no es editable ni es posible interactuar con él. Acepta los valores **true o false**, siendo false su valor por defecto:

```
<div id="elemento-clicable" aria-disabled="true"></div>
```

⇒ ***aria-grabbed***. Este atributo se usa con aquellos **elementos que es posible arrastrar y soltar**, **indicando si un elemento ha sido agarrado o no**. De estarlo, el atributo *aria-grabbed* tendrá el valor true, o false en otro caso:

```
<ul>
  <li id="el-1" aria-grabbed="true">Elemento 1</li>
  <li id="el-2" aria-grabbed="false">Elemento 2</li>
</ul>
```

⇒ *aria-dropeffect*. Se usa con aquellos **elementos que es posible arrastrar y soltar**, e indica el resultado que se obtiene cuando se suelta el elemento en otro elemento que lo acepte. Se aceptan uno o más valores, estando en este último caso separados por espacios. Los posibles **valores** son los siguientes:

- **copy**: Se creará una copia del elemento original y será copiada en el destino al soltar el elemento.
- **execute**: Se ejecutará una función que soporte el elemento de destino cuando se suelte el elemento, usando el elemento original como parámetro de dicha función.
- **link**: Se creará una referencia al elemento original en el destino al soltar el elemento.
- **move**: El elemento original se eliminará del contenedor origen y se moverá el contenedor de destino.
- **none**: No sucederá nada, por lo que se cancelará la acción.
- **popup**: Se mostrará una ventana emergente o popup que permitirá al usuario seleccionar entre alguna de las opciones anteriores, además de poder cancelar el evento.

El valor por defecto del atributo *aria-dropeffect* es **none**.

```
<ul">
  <li id="el-1" aria-dropeffect="move>Elemento 1</li>
  <li id="el-2" aria-dropeffect="move>Elemento 2</li>
</ul>
```

⇒ ***aria-expanded***: Este atributo se utiliza para indicar si un elemento o conjunto de elementos están **expandidos o colapsados**. Suele usarse, por ejemplo, en los submenús de los menús desplegables. Acepta los valores **true, false o undefined**, siendo este último su valor por defecto:

```
<ul role="navigation>
  <li><a href="/inicio">Inicio</a></li>
  <li>
    <a href="/informacion">Información</a>
    <ul class="nav-submenu" aria-expanded="true">
      <li><a href="/programacion">Programación</a></li>
      <li><a href="/aplicaciones">Diseño</a></li>
    </ul>
  </li>
</ul>
```

⇒ ***aria-haspopup***. Este atributo se usa para indicar que un elemento dispone de un **popup o submenú relacionado que se activará al interactuar con el elemento**. Los tooltips no se cuentan como popups:

```
<div aria-haspopup="true" id="subir-archivo">Archivo</div>
```

⇒ ***aria-invalid***. Este atributo se usa para indicar que el **valor introducido por el usuario en el elemento actual no es válido**. El lector de texto o navegador debe informar al usuario del error, además de proporcionar sugerencias para las posibles correcciones. El atributo solamente ha de estar presente si se han introducido datos en el elemento o campo, salvo que el usuario haya intentado enviar un formulario y el campo use el atributo *aria-required*, en cuyo caso, el atributo *aria-invalid* podrá tener un valor, aunque no se hayan introducido datos. Acepta los siguientes **valores**:

- **grammar**: se ha detectado un error gramatical en el valor.
- **spelling**: se ha detectado un error ortográfico en el valor.
- **true:** el valor introducido ha fallado la validación.

- **false**: no se han detectado errores en el valor.

El valor por defecto del atributo *aria-invalid* es **false**. A continuación, se muestra un ejemplo de uso:

```
<input type="test" aria-invalid="true"/>
```

⇒ ***aria-live***. Este atributo se usa para indicar que **el contenido de un elemento se modifica o actualiza de forma dinámica** cada cierto tiempo sin necesidad de que intervenga el usuario. Acepta los siguientes **valores**:

- **off**: no se alertará al usuario de los cambios. Esta opción suele utilizarse cuando el contenido de la actualización es poco relevante.
- **polite**: se alertará al usuario de los cambios, pero sin interrumpir su tarea actual. Por ejemplo, se alertará al usuario cuando termine de escribir o de leer algún otro contenido.
- **assertive:** se alertará al usuario de los cambios inmediatamente, independientemente de que hayan terminado su tarea actual o no. Este valor debe restringirse a avisos muy importantes.

El valor por **defecto** del atributo *aria-live* es **off**. A continuación, se muestra un ejemplo de uso:

```
<div role="region" id="informacion-coche" aria-live="polite">
  <h2 id="modelo">No se ha seleccionado nada</h2>
  <p id="descripcion-modelo">Selecciona un modelo para ver su descripción</p>
</div>
```

⇒ ***aria-atomic***. Con este atributo es posible especificar las **partes de un contenido dinámico que queremos que sean anunciadas al usuario**. Acepta los valores **true o false**, siendo false su valor por defecto. Si su valor es true, se anunciarán al usuario todos los elementos, hayan cambiado o no, pero si su valor

el false, se anunciarán al usuario exclusivamente aquellos elementos que hayan cambiado.

```
<ul id="noticias" aria-live="polite" aria-atomic="true">
  <li>Noticia 1</li>
  <li>Noticia 2</li>
  <li>Noticia 3</li>  </ul>
```

⇒ ***aria-autocomplete***. Este atributo sirve para **indicar si se proporcionan sugerencias escritura** para un elemento. Acepta los siguientes **valores**:

- **both**: se mostrará una lista de posibles sugerencias, además de aquella que está seleccionada.
- **inline**: se mostrará un texto con una sugerencia acerca de cómo rellenar el campo.
- **list**: se mostrarán una lista de elecciones de entre las cuales los usuarios podrán seleccionar una.
- **none**: no se mostrarán suferencias de escritura.

El valor por defecto del atributo *aria-autocomplete* es **none**. A continuación se muestra un ejemplo de uso:

```
<input type="text" aria-readonly="true" aria-owns="lista-ciudades"
aria-autocomplete="list" role="combobox" id="seleccion-ciudad">
<ul aria-expanded="true" role="listbox" id="lista-ciudades">
  <li role="option" id="opcion-1">Madrid</li>
  <li role="option" id="opcion-2">Barcelona</li>
</ul>
```

⇒ ***aria-owns***. Se utiliza para identificar una **relación de parentesco visual o contextual**. Acepta como parámetro el id o los ids de los elementos que pertenecen al elemento actual:

```
<input type="text" aria-owns="lista-paises">
<ul aria-expanded="true" role="listbox" id="lista-paises">
...
</ul>
```

⇒ ***aria-posinset***. Define la **posición numérica de un elemento en la lista de elementos actual**. Si todos los elementos están ya presentados en el DOM, este atributo no es necesario, que el lector o el navegador es capaz de deducir la posición de cada elemento. Sin embargo, si solamente se muestran por pantalla algunos elementos de la lista, esta propiedad sí es necesaria. Este atributo acepta un número entero como valor:

```
<h2 id="frutas">Frutas</h2>
<ul role="listbox" aria-labelledby="frutas">
  <li role="option" aria-setsize="4" aria-posinset="2">Piña</li>
  <li role="option" aria-setsize="4" aria-
posinset="3">Naranja</li>
</ul>
```

⇒ ***aria-setsize***. Sirve para **indicar el número de elementos total de una lista** de elementos. Este atributo suele usarse **junto** con el atributo *aria-posinset*:

```
<h2 id="colores">Colores</h2>
<ul role="listbox" aria-labelledby="colores">
  <li role="option" aria-setsize="16" aria-posinset="2">Verde</li>
  <li role="option" aria-setsize="16" aria-posinset="3">Rojo</li>
  <li role="option" aria-setsize="16" aria-posinset="4">Azul</li>
</ul>
```

⇒ ***aria-selected***. Este atributo se usa para indicar **si un elemento está o no seleccionado**. Es **similar** a los atributos *aria-checked* o *aria-pressed*. Se suele usar con **widgets** de selección simple o múltiple. Acepta el valor true si el elemento está seleccionado o false si no lo está:

```
<ul>
  <li role="tab" id="elemento-a"
aria-selected="true">Selector 1</li>
  <li role="tab" id="elemento-b"
aria-selected="false">Selector 2</li>
</ul>
```

⇒ ***aria-sort***. Este atributo se usa para indicar **si los elementos de una tabla o cuadrícula están ordenados en orden ascendente o descendente**. Esta propiedad solamente se debe aplicar a las cabeceras de las tablas o de los elementos grid. Acepta los siguientes **valores**:

- **ascending**: los elementos se ordenan en orden ascendente.
- **descending**: los elementos se ordenan en orden descendente.
- **other:** se utiliza otro algoritmo de ordenación.
- **none**: no existe un orden definido, siendo este el valor por defecto.

Ejemplo de uso del atributo ***aria-sort***:

```
<table role="grid">
  <thead>
    <tr role="row">
      <th scope="col" role="columnheader" aria-
sort="ascending">Fruta▲/th>
      <th scope="col" role="columnheader">Cantidad</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td>Naranja</td>
      <td>6</td>
    </tr>
    <tr>
      <td>Manzana</td>
```

```
      <td>4</td>
    </tr>
    <tr>
      <td>Pera</td>
      <td>2</td>
    </tr>
  </tbody>
</table>
```

⇒ ***aria-valuenow***. En ocasiones existen elementos o **widgets que permiten seleccionar un valor** de entre un rango de valores. Suelen estar representados por sliders o barras de progreso. Este atributo permite **establecer el valor actual o el valor por defecto**, de haberlo, de estos elementos:

```
<div role="slider" aria-valuenow="20"></div>
```

⇒ ***aria-valuemin***. Permite establecer el **valor mínimo de un elemento que permita seleccionar valores**, como un slider o barra de progreso:

```
<div role="slider" aria-valuenow="20" aria-valuemin="1"
aria-valuemax="100"></div>
```

⇒ ***aria-valuemax***. Permite establecer el **valor máximo de un elemento que permita seleccionar valores**, como un slider o barra de progreso:

```
<div role="slider" aria-valuenow="20" aria-valuemin="1"
aria-valuemax="100"></div>
```

⇒ ***aria-valuetext***. Permite **establecer el valor en forma de texto legible** de un elemento que permita seleccionar valores. Si este atributo está presente, también debería estarlo el atributo *aria-valuenow*:

```
<div role="slider" aria-valuenow="20" aria-valuetext="perro"></div>
```

3.6 Tabla de puntos de verificación

Partiendo de las **Pautas de Accesibilidad para el Contenido en la Web**, los diseñadores, desarrolladores o evaluadores que verifican y miden el nivel de accesibilidad en un sitio web, utilizan una tabla de **puntos de verificación de accesibilidad** para garantizar que el contenido sea accesible para todas las personas.

Cada punto de verificación corresponde a un aspecto clave de la accesibilidad, como proporcionar texto alternativo para imágenes, asegurar que el contenido sea navegable mediante el teclado, evitar parpadeos que puedan causar convulsiones, entre otros.

Cada desarrollador puede crear su propia tabla de puntos de verificación. En el siguiente enlace se muestra un ejemplo de tabla de puntos de verificación (en inglés). Esta tabla contiene una columna con los puntos a verificar y 3 columnas más (Sí, No, N/A) para indicar si se cumple la verificación. **w3.org/TR/WAI-WEBCONTENT/full-checklist**

4. Herramientas para la evaluación de accesibilidad

Existen varias herramientas de evaluación, análisis y testeo de accesibilidad web que pueden ayudarte a determinar si el contenido web cumple con los estándares de accesibilidad. Algunas de estas herramientas se pueden encontrar en la página oficial, en el siguiente enlace: **w3.org/WAI/test-evaluate/es**. Este listado incluye información sobre más de 100 herramientas. Además, tiene filtros para limitar la lista a los tipos de herramientas que nos interesen.

El **proceso de evaluación** de la accesibilidad mediante herramientas, se divide en **dos fases**:

1) En primer lugar, se debe realizar un **análisis automático** que detecte los problemas de accesibilidad;

2) En segundo lugar, debe llevarse a cabo una **evaluación manual** para identificar todos aquellos problemas que no pueden ser comprobados en la primera fase.

4.1 Herramientas online

En cuanto a las herramientas automáticas, algunas de las más reconocidas en la actualidad son:

- **Validador HTML de W3C**. Realiza una validación de la gramática del sitio web, comprobando la conformidad de los documentos HTML con respecto a las gramáticas del W3C.

- **Validador de CSS de W3C**. Realiza una validación de la gramática centrada en la verificación de las hojas de estilo CSS y otros documentos HTML con hojas de estilo.

- **Diversas aplicaciones online** para evaluar y validar la accesibilidad de un sitio web como:
  - **Taw**. Aplicación web que permite hacer un análisis de accesibilidad de forma gratuita tan solo introduciendo la URL ().

  - **Accessibilitychecker.org**. Informe bastante completo sobre accesibilidad.

- **Color Contrast Checker:** aplicación para **comprobar el contraste** entre fondo y texto.

- **Colour Constrast Analyser**: aplicación para comprobar el contraste entre fondos y textos (sólo para mac y windows).

- **WAVE**. Aplicación online que permite evaluar la accesibilidad de un sitio web completo.

- **Textise**: simulador online para ver una página web en modo "Sólo texto".

- **WAI HTML Table Linearizar Entry Form**: aplicación para **evaluar tablas**.

- **Powermapper**. Aplicación online que chequea y valida la accesibilidad de una página web. En estos chequeos se incluye la WCAG 2.0, 2.1 y 2.2.

- **Observatorio de Accesibilidad Web** de la Universidad Politécnica Salesiana de Ecuador.

- **Mauve++**. Es un validador de accesibilidad web, que soporta WCAG 2.0 y 2.1.

- **JAWS**: lector de pantalla (sistemas operativos Windows).

4.2 Extensiones para navegadores web

La mayoría de los **navegadores** permiten facilitar el proceso de evaluación de la accesibilidad de una web de forma manual mediante una serie de extensiones que permiten, entre otras cosas, modificar la resolución de forma rápida, comprobar los colores de un sitio web, etcétera. Algunas de estas **extensiones**, pueden ser:

- **Web Developer**: es una barra de herramientas que se utiliza **para Chrome y Mozilla Firefox** con las diferentes opciones que pueden utilizar los desarrolladores web.

- **Firefox WAVE Accessibility Extension**: extensión para la barra de herramientas utilizada en **Chrome y Mozilla Firefox** con las diferentes opciones que facilitan la navegación por los diferentes contenidos de aquellos usuarios con alguna discapacidad.

- **AInspector 3.0**. Extensión para Firefox, que inspecciona las páginas web en busca de características y problemas relacionados con los requisitos WCAG 2.2 en los niveles A y AA.

- **Axe DevTools.** Extensión para chequear la accesibilidad de un sitio web, para Chrome.

- **Mauve++**. Es la extensión equivalente a su aplicación online.

- **Web Accessibility Toolbar**: plugin utilizado en **Internet Explorer** que permite una evaluación manual sencilla de la accesibilidad de un sitio web.

- **Firebug**: extensión de Firefox que permite que los desarrolladores puedan realizar modificaciones sobre el código fuente HTML, CSS, Javascript, etc.

- **HTML Validator Tidy**: extensión para Mozilla que permite agregar un validador de HTML en el interior de Firefox.

- **Helperbird Toolbar**. Extensión para Chrome y Firefox que nos permite personalizar la web según necesidades para mejorar la accesibilidad, la lectura, la escritura, el estudio y la productividad, entre otros.

- **Screen Reader**. Extensión para Chrome que simula un lector de pantalla para la web.

5. Guías prácticas

En esta sección se presentarán algunas guías prácticas de cómo implementar ciertos elementos para que sean accesibles.

5.1 Cómo hacer imágenes accesibles

Hacer imágenes accesibles es fundamental para garantizar que todas las personas, incluyendo aquellas con barreras visuales o cognitivas, puedan comprender el contenido de las imágenes en tu sitio web.

A continuación, se presentan algunas prácticas clave para hacer imágenes accesibles:

1) **Agregar texto alternativo (alt text)**. El atributo ***alt*** proporciona un texto alternativo que se muestra cuando la imagen no puede cargarse o no es visible para el usuario. Debes utilizar una descripción concisa y significativa que transmita el propósito o contenido de la imagen. **Ejemplo**:

```
<img src="imagen.jpg" alt="Grupo de amigos disfrutando de un día soleado en el parque">
```

2) **Usar imágenes decorativas**. Si la imagen no aporta información relevante al contenido o es puramente decorativa, es mejor usar un atributo ***alt*** **vacío para indicar a los lectores de pantalla que la imagen no es relevante** para la comprensión del contenido. **Ejemplo**:

```
<img src="decoracion.png" alt="">
```

3) **Tamaño y formato adecuado.** Asegúrate de **optimizar** el tamaño y formato de las imágenes para que carguen rápidamente en la página. Además, evita utilizar imágenes con texto incrustado, ya que pueden resultar poco legibles para personas con discapacidades visuales.

4) **Textos descriptivos cercanos**. Si una imagen es relevante para el contenido cercano, es útil incluir el texto descriptivo cerca de la imagen en el código HTML. Esto ayuda a los lectores de pantalla a asociar la imagen con el texto. **Ejemplo**:

```
<p>El nuevo edificio de la biblioteca <img src="biblioteca.jpg" alt="Fachada del nuevo edificio de la biblioteca"> es un lugar acogedor y moderno para estudiar.</p>
```

5) **Evitar información importante en imágenes.** No coloques información importante, como texto crítico o enlaces, únicamente en una imagen. Es mejor usar elementos de texto reales en HTML para garantizar que todos los usuarios puedan acceder a la información. **Ejemplo**: no incluyas una imagen con el número de teléfono sin indicar texto alternativo porque los usuarios con problemas visuales no podrían llamarte.

6) **Contraste de color**. Si la imagen contiene texto, asegúrate de que haya suficiente contraste entre el texto y el fondo para que sea legible para todas las personas.

7) **Agregar título (title).** Si es necesario, puedes utilizar el atributo ***title*** para proporcionar información adicional sobre la imagen, pero evita duplicar la información del texto alternativo. **Ejemplo**:

```
<img src="proyecto.jpg" alt="Proyecto de construcción" title="Vista panorámica del proyecto de construcción">
```

5.2 Cómo hacer enlaces accesibles

Para hacer enlaces accesibles en una página web, es importante seguir algunas pautas que mejoren la experiencia para todos los usuarios, incluyendo aquellos con algún tipo de barrera.

Algunas recomendaciones para hacer enlaces accesibles son las siguientes:

1) **Utilizar texto descriptivo**. En lugar de colocar enlaces como “haz clic aquí” o “más información”, **utiliza texto que describa a dónde llevará el enlace**, como “Leer más sobre nuestro equipo” o “Visitar nuestra página de contacto”.

   Enlace no accesible:

```
<a href="about.html">Haz clic aquí</a>
```

Enlace accesible:

```
<a href="contact.html" title="Visita nuestra página de contacto">Contáctanos</a>
```

2) **Añadir atributo "title"**. Proporciona un texto adicional a través del atributo "title" para ofrecer más información sobre el enlace cuando se pasa el cursor sobre él.

```
<a href="contact.html" title="Visita nuestra página de contacto">Contáctanos</a>
```

3) **No dependas solo del color.** Asegúrate de que los enlaces sean **distinguibles** sin depender únicamente del color. Utiliza subrayados, negritas o diferentes estilos para destacarlos.

Enlace no accesible:

```
<a href="services.html" style="color: blue;">Servicios</a>
```

Enlace accesible:

```
<a href="services.html" style="color: blue; text-decoration: underline;">Servicios</a>
```

4) **Evitar enlaces vacíos.** Evita utilizar enlaces sin contenido, como "#", ya que esto puede ser confuso para los lectores de pantalla.

```
<a href="#">Volver al inicio</a>
```

5) **Asegurar un orden lógico.** Organiza tus enlaces de manera coherente y en un orden lógico para facilitar la navegación.

```
<nav>
  <ul>
    <li><a href="home.html">Inicio</a></li>
    <li><a href="about.html">Sobre nosotros</a></li>
    <li><a href="services.html">Servicios</a></li>
    <li><a href="contact.html">Contacto</a></li>
  </ul>
</nav>
```

6) **Considerar el tamaño de los enlaces**. Asegúrate de que los enlaces sean lo suficientemente grandes como para ser seleccionados fácilmente en dispositivos táctiles.

7) **Probar con lectores de pantalla**. Verifica la accesibilidad de los enlaces utilizando lectores de pantalla para asegurarte de que la experiencia sea óptima para todos los usuarios.

Siguiendo estas recomendaciones, estarás contribuyendo a que tu sitio web sea más inclusivo y accesible para todas las personas.

5.3 Cómo hacer tablas accesibles

La importancia de hacer **tablas accesibles** radica en garantizar que **todas las personas puedan acceder y comprender la información presentada** en las tablas. Esto mejora la experiencia de usuario, fomenta la inclusión y cumple con los estándares de accesibilidad web, promoviendo un entorno digital más igualitario y accesible.

Además, las tablas accesibles **también benefician a los motores de búsqueda**, ya que facilitan la indexación y comprensión del contenido, mejorando el posicionamiento en los resultados de búsqueda.

Las tablas son una herramienta útil para presentar datos en un formato estructurado y fácil de entender. Sin embargo, para asegurarnos de que nuestras tablas sean accesibles para todos los usuarios, es importante seguir las siguientes pautas de accesibilidad.

5.3.1 Utilizar encabezados de tabla

Los encabezados de tabla ayudan a los usuarios a entender la estructura y el contenido de la tabla. Asegúrate de utilizar la etiqueta `<th>` para los **encabezados de tabla** y de **incluir un atributo** `scope` para indicar si el encabezado se aplica a una columna (`scope="col"`) o a una fila (`scope="row"`). Esto mejorará la interpretación de la tabla por parte de los lectores de pantalla.

```
<table>
  <tr>
    <th scope="col">Nombre</th>
    <th scope="col">Edad</th>
    <th scope="col">Ciudad</th>
  </tr>
  <tr>
    <td>Juan</td>
    <td>32</td>
    <td>Madrid</td>
  </tr>
  <tr>
    <td>Andrea</td>
    <td>27</td>
    <td>Elche</td>
  </tr>
</table>
```

5.3.2 Proporcionar una descripción

Si tu tabla es compleja o contiene una gran cantidad de información, puede ser útil proporcionar una descripción para ayudar a los usuarios a entender su contenido. Puedes utilizar la etiqueta `<caption>` dentro de la etiqueta `<table>` para agregar una descripción a tu tabla.

```
<table>
  <caption>Lista de participantes</caption>
  <tr>
    <th scope="col">Nombre</th>
    <th scope="col">Edad</th>
    <th scope="col">Ciudad</th>
  </tr>
  <tr>
    <td>Juan</td>
    <td>32</td>
    <td>Madrid</td>
  </tr>
  <tr>
    <td>Andrea</td>
    <td>27</td>
    <td>Elche</td>
  </tr>
</table>
```

5.3.3 Utilizar un diseño claro y sencillo

Un diseño claro y sencillo ayuda a los usuarios a entender el contenido de la tabla. **Evita utilizar colores o fuentes que puedan dificultar la lectura** y asegúrate de que el **contenido esté bien organizado** y sea fácil de seguir.

5.3.4 Usar el atributo headers

El atributo ***headers*** se puede utilizar para asociar celdas de datos con celdas de encabezado en tablas complejas. Para ello, debemos asociar las celdas de datos (***<td>***) con los encabezados (***<th>***) mediante el atributo ***headers***. Esto puede ayudar a mejorar la accesibilidad para usuarios que utilizan lectores de pantalla.

```
<table>
  <tr>
    <th id="nombre">Nombre</th>
    <th id="edad">Edad</th>
    <th id="ciudad">Ciudad</th>
  </tr>
  <tr>
    <td headers="nombre">Juan</td>
    <td headers="edad">32</td>
    <td headers="ciudad">Madrid</td>
  </tr>
  <tr>
    <td headers="nombre">Andrea</td>
    <td headers="edad">27</td>
    <td headers="ciudad">Elche</td>
  </tr>
</table>
```

En este ejemplo, hemos utilizado el atributo ***headers*** para asociar cada celda de datos con su celda de encabezado correspondiente. Esto puede ayudar a los usuarios que utilizan lectores de pantalla a entender mejor la estructura y el contenido de la tabla.

5.3.5 Agregar etiquetas <thead>, <tbody> y <tfoot>

Las etiquetas ***<thead>***, ***<tbody>*** y ***<tfoot>*** se pueden utilizar para mejorar la estructura y la accesibilidad de tus tablas. La etiqueta ***<thead>*** se utiliza para agrupar el contenido del encabezado de la tabla, la etiqueta ***<tbody>*** se utiliza para agrupar el contenido principal de la tabla y la etiqueta ***<tfoot>*** se utiliza para agrupar el contenido del pie de la tabla.

```
<table>
  <thead>
    <tr>
      <th>Nombre</th>
      <th>Edad</th>
      <th>Ciudad</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td>Juan</td>
      <td>32</td>
      <td>Madrid</td>
    </tr>
    <tr>
      <td>Andrea</td>
      <td>27</td>
      <td>Elche</td>
    </tr>
  </tbody>
  <tfoot>
    <tr>
      <td colspan="3">Total: 2 participantes</td>
    </tr>
  </tfoot>
</table>
```

En este ejemplo, hemos utilizado las etiquetas **`<thead>`**, **`<tbody>`** y **`<tfoot>`** para mejorar la estructura y la accesibilidad de nuestra tabla. Esto puede ayudar a los usuarios a entender mejor el contenido de la tabla y también puede mejorar la compatibilidad con lectores de pantalla.

5.3.6 Evitar celdas vacías o duplicadas

Las celdas vacías o duplicadas pueden dificultar la comprensión de la tabla para los usuarios, especialmente para aquellos que utilizan lectores de pantalla. Asegúrate de que todas las celdas de tu tabla contengan contenido relevante y evita duplicar información.

Si necesitas fusionar celdas en una fila o columna, puedes utilizar los atributos ***colspan*** y ***rowspan*** para hacerlo. Estos atributos te permiten especificar cuántas columnas o filas debe ocupar una celda.

```
<table>
  <tr>
    <th>Nombre</th>
    <th>Edad</th>
    <th>Ciudad</th>
  </tr>
  <tr>
    <td>Juan</td>
    <td>32</td>
    <td>Madrid</td>
  </tr>
  <tr>
    <td>Andrea</td>
    <td>27</td>
    <td>Elche</td>
  </tr>
  <tr>
    <td colspan="3">Total: 2 participantes</td>
  </tr>
</table>
```

En este ejemplo, hemos utilizado el atributo ***colspan*** para fusionar las tres celdas de la última fila en una sola celda. Esto nos permite mostrar un resumen del contenido de la tabla sin dejar celdas vacías o duplicar información.

5.3.7 Proporcionar información adicional con <abbr> y el atributo title

La etiqueta ***<abbr>*** se utiliza para indicar una abreviatura y el atributo ***title*** se utiliza para proporcionar información adicional sobre el contenido de la etiqueta.

```
<table>
  <tr>
    <th>Nombre</th>
    <th>Edad</th>
    <th><abbr title="Ciudad de residencia">Ciudad</abbr></th>
  </tr>
  <tr>
    <td>Juan</td>
    <td>32</td>
    <td>Madrid</td>
  </tr>
  <tr>
    <td>María</td>
    <td>27</td>
    <td>Barcelona</td>
  </tr>
</table>
```

En este ejemplo, hemos utilizado la etiqueta `<abbr>` para indicar que la palabra «Ciudad» es una abreviatura de "Ciudad de residencia". También hemos utilizado el atributo `title` para proporcionar información adicional sobre el contenido de la celda.

5.3.8 Hacer que la tabla sea responsive

Hacer que tu tabla sea responsive, es decir, que se adapte al tamaño de la pantalla del dispositivo del usuario, puede mejorar significativamente su accesibilidad. Una tabla responsive se ajusta automáticamente para mostrar su contenido de manera clara y legible en pantallas de diferentes tamaños, lo que facilita su uso para todos los usuarios.

Hay varias **técnicas** que puedes utilizar para hacer que tus tablas sean responsive. Una opción es utilizar las **media queries** (`@media`) para aplicar diferentes estilos a tu tabla según el tamaño de la pantalla del dispositivo. Otra opción es utilizar un diseño de cuadrícula flexible (`flex o grid`) para permitir que las celdas de la tabla se ajusten automáticamente al tamaño de la pantalla. También puedes utilizar la propiedad `overflow-x: auto;` para permitir el desplazamiento horizontal en tablas grandes en dispositivos pequeños.

Siguiendo estas pautas, podrás hacer que tus tablas sean accesibles y ofrecer una mejor experiencia a todos los usuarios. Además, no te olvides de revisar el contraste entre el texto y el fondo y de realizar pruebas de accesibilidad para asegurarte de que la tabla sea accesible.

5.3.9 Ejemplos

Ejemplo 1:

Esta tabla representa una lista de productos con sus precios y disponibilidad en euros. Cada columna está correctamente etiquetada con el atributo ***scope="col"***, y se han añadido los atributos ***id*** en las celdas de cabecera correspondientes. Las celdas de datos están etiquetadas con el atributo ***scope="row"*** para asociarlas con la cabecera de la columna correspondiente. Además, se ha añadido un pie de tabla (***<tfoot>***) que indica que los precios están expresados en euros.

Código HTML:

```
<table>
  <caption>Tabla de precios de productos</caption>
  <thead>
    <tr>
      <th scope="col">Producto</th>
      <th scope="col" id="precio-producto">Precio</th>
      <th scope="col" id="disponibilidad-producto">Disponibilidad</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <th scope="row">Producto 1</th>
      <td headers="precio-producto">20,00 €</td>
      <td headers="disponibilidad-producto">En stock</td>
    </tr>
    <tr>
      <th scope="row">Producto 2</th>
      <td headers="precio-producto">25,00 €</td>
      <td headers="disponibilidad-producto">Agotado</td>
    </tr>
  </tbody>
  <tfoot>
    <tr>
      <td colspan="3">Precios en euros.</td>
    </tr>
  </tfoot>
</table>
```

Código CSS:

```css
/* Estilos para la tabla */
.prices-table {
  width: 100%;
  border-collapse: collapse;
}

/* Estilos para el encabezado de la tabla */
.prices-table th {
  background-color: #f2f2f2;
  border: 1px solid #ddd;
  padding: 8px;
}

/* Estilos para las celdas de datos */
.prices-table td {
  border: 1px solid #ddd;
  padding: 8px;
}

/* Estilos para el pie de tabla */
.prices-table tfoot td {
  background-color: #f2f2f2;
  border: 1px solid #ddd;
  padding: 8px;
  font-weight: bold;
  text-align:center;
}
```

Tabla 1. Precios de productos

Producto	Precio	Disponibilidad
Producto 1	20,00 €	En stock
Producto 2	25,00 €	Agotado
Precios en euros.		

Ejemplo 2:

En este ejemplo, la tabla muestra información sobre tres estudiantes y sus calificaciones en diferentes asignaturas. Los estilos aplicados son similares al ejemplo anterior, con bordes y relleno en las celdas para mejorar la legibilidad. Además, se utiliza el atributo ***scope*** para indicar el alcance de las celdas de encabezado y se agrega un pie de tabla para proporcionar información adicional sobre las calificaciones.

Además, se utilizan los elementos y atributos HTML adecuados, como ***caption***, ***thead***, ***tbody***, ***tfoot***, ***th***, y ***scope***, para mejorar la estructura y la navegación para usuarios con discapacidades visuales o que utilizan lectores de pantalla.

Código HTML:

```
<table class="grades-table">
  <caption><b>Tabla 2.</b> Calificaciones de estudiantes</caption>
  <thead>
    <tr>
      <th scope="col">Nombre</th>
      <th scope="col">Asignatura</th>
      <th scope="col">Calificación</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <th scope="row">Juan Pérez</th>
      <td>Lengua y Literatura</td>
      <td>8.5</td>
    </tr>
    <tr>
      <th scope="row">María Gómez</th>
      <td>Matemáticas</td>
      <td>9.2</td>
    </tr>
    <tr>
      <th scope="row">Carlos López</th>
      <td>Ciencias Sociales</td>
      <td>7.8</td>
    </tr>
  </tbody>
  <tfoot>
    <tr>
      <td colspan="3">Calificaciones en escala de 0 a 10.</td>
    </tr>
  </tfoot>
</table>
```

Código CSS:

```css
table{caption-side: bottom;}
/* Estilos para la tabla */
.grades-table {
  width: 100%;
  border-collapse: collapse;
}

/* Estilos para el encabezado de la tabla */
.grades-table th {
  background-color: #f2f2f2;
  border: 1px solid #ddd;
  padding: 8px;
}

/* Estilos para las celdas de datos */
.grades-table td {
  border: 1px solid #ddd;
  padding: 8px;
}

/* Estilos para el pie de tabla */
.grades-table tfoot td {
  background-color: #f2f2f2;
  border: 1px solid #ddd;
  padding: 8px;
  font-weight: bold;
  text-align:center;
}
```

Nombre	Asignatura	Calificación
Juan Pérez	Lengua y Literatura	8.5
María Gómez	Matemáticas	9.2
Carlos López	Ciencias Sociales	7.8
Calificaciones en escala de 0 a 10.		

Tabla 2. Calificaciones de estudiantes

5.4 Cómo hacer formularios accesibles

Los **formularios** son elementos fundamentales en muchos sitios web, ya que permiten a los usuarios enviar información y realizar acciones. Para garantizar formularios accesibles, es importante seguir ciertas pautas y buenas prácticas.

A continuación, se presentan algunas técnicas **HTML** con ejemplos que nos ayudarán a crear formularios accesibles.

5.4.1 Etiquetas y atributos

Utiliza etiquetas `<label>` para cada campo de entrada y asocia cada etiqueta con su respectivo campo utilizando el atributo `for` o anidando el campo dentro de la etiqueta `<label>`. **Ejemplo**:

```
<form>
    <!-- Etiqueta asociada utilizando el atributo 'for' -->
    <label for="nombre">Nombre:</label>
    <input type="text" id="nombre" name="nombre" required>

    <!-- Etiqueta anidada dentro del elemento 'label' -->
    <label>
        Apellido:
        <input type="text" id="apellido" name="apellido" required>
    </label>
    <input type="submit" value="Enviar">
</form>
```

También es recomendable utilizar el atributo `aria-label` para proporcionar etiquetas accesibles en caso de que las etiquetas visibles no sean suficientemente descriptivas. **Ejemplo**:

```
<button aria-label="Cerrar ventana" onclick="cerrarVentana()">X</button>
```

En este ejemplo, hemos creado un botón con una “X” que representa un botón de cerrar ventana. Sin embargo, como no se proporciona un nombre entendible de la acción, hemos utilizado el atributo `aria-label` para proporcionar una etiqueta accesible para el botón.

Cuando un lector de pantalla o una herramienta de asistencia encuentre este botón, leerá el contenido del atributo `aria-label`, que en este caso es “Cerrar ventana”. Esto permite a los usuarios de asistencia comprender el propósito y la función del botón, incluso si no pueden ver la “X” visible.

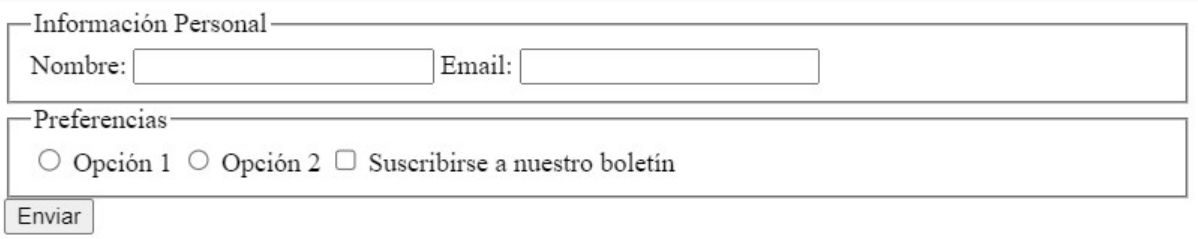
Es importante **utilizar** `aria-label` **con moderación y solo cuando sea necesario** para mejorar la accesibilidad de la interfaz para usuarios con discapacidades visuales o cognitivas.

5.4.2 Agrupación de campos

Utiliza etiquetas `<fieldset>` y `<legend>` para agrupar campos relacionados y proporcionar una estructura clara. Esto también es útil para usuarios de lectores de pantalla, ya que les permite navegar y comprender el formulario de manera más eficiente. **Ejemplo**:

```
<form>
  <fieldset>
    <legend>Información Personal</legend>
    <label for="nombre">Nombre:</label>
    <input type="text" id="nombre" required>
    <label for="email">Email:</label>
    <input type="email" id="email" required>
  </fieldset>

  <fieldset>
    <legend>Preferencias</legend>
    <label>
      <input type="radio" name="preferencia" value="opcion1" required>
      Opción 1
    </label>
    <label>
      <input type="radio" name="preferencia" value="opcion2" required>
      Opción 2
    </label>
    <label>
      <input type="checkbox" name="suscripcion" value="si">
      Suscribirse a nuestro boletín
    </label>
  </fieldset>
  <button type="submit">Enviar</button>
</form>
```



En este ejemplo, hemos dividido el formulario en dos grupos de campos relacionados utilizando la etiqueta `<fieldset>`. Cada `<fieldset>` contiene un `<legend>` que describe el grupo de campos, proporcionando una estructura clara para el formulario.

El primer `<fieldset>` agrupa los campos de «Información Personal», que incluye los campos de “Nombre” y “Email”. El segundo `<fieldset>` agrupa los campos de “Preferencias”, que incluye dos opciones de radio y una casilla de verificación.

Esto es útil para los usuarios de lectores de pantalla, ya que les permite navegar entre los grupos de campos de manera más eficiente y comprender cómo están organizados los campos en el formulario.

Además, hemos agregado el atributo `required` a algunos campos para indicar que son obligatorios. Esto asegura que los usuarios no puedan enviar el formulario sin completar estos campos requeridos.

5.4.3 Etiquetas para botones

Asegúrate de que los **botones tengan etiquetas descriptivas y significativas**. Evita utilizar simplemente “Enviar” o “Enviar formulario” como texto para el botón de envío, en su lugar, utiliza algo más descriptivo, como “Enviar consulta” o “Registrarse”.

```
<button type="submit" aria-label="Enviar consulta">Enviar</button>
```

5.4.4 Accesibilidad del teclado

Asegúrate de que los c**ampos de entrada y los botones sean totalmente navegables y operables mediante el teclado**. Esto es esencial para usuarios que no pueden utilizar un ratón. **Ejemplo**:

```
<form>
    <label for="nombre">Nombre:</label>
    <input type="text" id="nombre" required>

    <label for="email">Email:</label>
    <input type="email" id="email" required>

    <label for="consulta">Consulta:</label>
    <textarea id="consulta" required></textarea>

    <button type="submit" aria-label="Enviar consulta">Enviar</button>
    <!-- El botón de envío tiene una etiqueta descriptiva "Enviar
    consulta" utilizando el atributo aria-label -->
</form>
```

En este ejemplo, hemos agregado un formulario con campos de entrada de texto y un botón de envío.

Para asegurarnos de que los campos de entrada y los botones sean navegables y operables mediante el teclado, hemos utilizado etiquetas `<label>` para asociar cada campo de entrada con su respectiva etiqueta. Esto permite que los usuarios puedan seleccionar los campos de entrada utilizando la tecla “Tab” y navegar entre ellos utilizando las teclas de flecha.

También hemos agregado el atributo `aria-label` al botón de envío para proporcionar una etiqueta descriptiva “Enviar consulta”. Esto asegura que los usuarios puedan entender la función del botón incluso cuando están navegando y operando el formulario mediante el teclado.

Es importante tener en cuenta la accesibilidad mediante teclado para garantizar que todos los usuarios, incluidos aquellos que no pueden utilizar un ratón, puedan interactuar con el formulario sin problemas. Esto mejora significativamente la experiencia del usuario y hace que el formulario sea más inclusivo para todas las personas.

5.4.5 Etiqueta de campo requerido

Utiliza el atributo ***required*** para indicar que ciertos campos son obligatorios y proporciona una etiqueta o texto indicativo para estos campos. **Ejemplo**:

```
<form>
  <label for="nombre">Nombre:</label>
  <input type="text" id="nombre" required>
  <span class="required-label">(Campo obligatorio)</span>

  <label for="email">Email:</label>
  <input type="email" id="email" required>
  <span class="required-label">(Campo obligatorio)</span>

  <button type="submit">Enviar</button>
</form>
```

5.4.6 Contraste de color, tamaño y formato

Utiliza colores y contrastes adecuados para que los campos y etiquetas **sean fácilmente legibles** para todos los usuarios, incluyendo aquellos con discapacidad visual. Asegúrate de que los **campos de entrada tengan un tamaño adecuado y un formato claro**. Puedes poner ejemplos para ayudar a los usuarios a completar el formulario correctamente.

5.4.7 Orden de navegación

Define un orden de navegación lógico para los campos de entrada, de modo que los usuarios puedan avanzar de manera coherente y predecible. En caso de necesitarlo, también se puede utilizar el atributo ***tabindex*** para establecer el orden de navegación. **Ejemplo**:

```
<form>
    <label for="nombre">Nombre:</label>
    <input type="text" id="nombre" tabindex="1" required>

    <label for="apellido">Apellido:</label>
    <input type="text" id="apellido" tabindex="2" required>

    <label for="email">Email:</label>
    <input type="email" id="email" tabindex="3" required>

    <label for="telefono">Teléfono:</label>
    <input type="tel" id="telefono" tabindex="4">

    <label for="direccion">Dirección:</label>
    <input type="text" id="direccion" tabindex="5">

    <label for="ciudad">Ciudad:</label>
    <input type="text" id="ciudad" tabindex="6">

    <label for="pais">País:</label>
    <input type="text" id="pais" tabindex="7">

    <button type="submit" tabindex="8">Enviar</button>
</form>
```

En este ejemplo, hemos utilizado el atributo ***tabindex*** para establecer el orden de navegación. Cada campo de entrada tiene un valor de ***tabindex*** que indica el orden en el que se puede acceder a ellos mediante la tecla ***Tab***. El valor más bajo tiene el enfoque primero, y el valor más alto es el último.

Hemos comenzado con el campo de nombre, seguido del apellido, el email y otros campos de contacto. Luego, hemos continuado con los campos de dirección y ciudad, y finalmente el campo de país y el botón de envío.

Este orden de navegación lógico facilita a los usuarios avanzar de un campo a otro de manera coherente y predecible, mejorando la experiencia de usuario en el formulario.