

Herramientas de Accesibilidad Web

2º DAW Diseño de Interfaces

IES Ribera de Castilla

Enrique García Sáez

09/02/2021

1. CONCEPTO DE ACCESIBILIDAD	3
2. HERRAMIENTAS DE DESARROLLADOR	3
Asociar un <label> a cada elemento de un formulario	3
Incluir alternativas para imágenes	4
Identificar el lenguaje de la página y configuración	4
Etiquetas semánticas y roles	5
Ayudar a los usuarios a evitar y corregir errores	6
Reflejar el orden de lectura en el orden del código	7
Código que se adapta a la tecnología del usuario	8
Proveer significado a los elementos interactivos no estándar	8
Asegurar que todos los elementos interactivos son accesibles desde el teclado	9
Evitar códigos CAPTCHA siempre que sea posible	9
3. HERRAMIENTAS DE DISEÑO	10
Proveer suficiente contraste entre el plano principal y el secundario	10
Utilizar diferenciaciones no basadas sólo en el color	10
Asegurar que los elementos interactivos son fáciles de identificar	10
Proveer opciones de navegación claras y consistentes	11
Aportar feedback fácilmente identificable	11
Utilizar encabezados y espacios para el contenido agrupado	11
Incluir alternativas para imágenes y otros contenidos de media en el diseño	11
Aportar controles para contenido que se ejecuta automáticamente	12
4. HERRAMIENTAS DE AUDITORÍA	12
5. Fuentes y recursos	14

1. CONCEPTO DE ACCESIBILIDAD

La accesibilidad es una característica que las interfaces web cumplen en mayor o menor grado. El objetivo inicial de su análisis es reducir las barreras de comunicación e interacción entre la interfaz y el usuario, considerando las limitaciones de personas discapacitadas y también necesidades de usuarios habituales.

Se dice que una web es accesible cuando está diseñada con herramientas para facilitar su acceso a usuarios con limitaciones, ya sean por discapacidades o por circunstancias especiales. Por ejemplo, un usuario sin discapacidad auditiva también se puede beneficiar de una web que utiliza subtítulos para los vídeos, ya que es posible que no pueda reproducir su sonido en un determinado contexto.

En los últimos años han aparecido herramientas que permiten a los desarrolladores adaptar las aplicaciones web a estas necesidades. Podemos diferenciar dos grandes grupos para mejorar la accesibilidad de la web: de desarrollador y de diseño.

2. HERRAMIENTAS DE DESARROLLADOR

Los siguientes aspectos o prácticas son algunas de las recomendadas por W3G, el consorcio que estandariza prácticas, lenguajes y conceptos para la World Wide Web. La mayoría de éstas están relacionadas con etiquetas html y cómo deben configurarse para que el navegador las renderice con el efecto deseado:

Asociar un <label> a cada elemento de un formulario

Asociar una etiqueta label a cada elemento del formulario permite que, al clicar sobre el label, se active el elemento de control del formulario (focus). Esto facilita la navegación por el formulario y reduce el tiempo del usuario para posicionarse sobre un input. Se consigue con el atributo “for” y con el valor del “id” del <input> correspondiente:

</> Code Snippet

```
<label for="username">Username</label>
<input id="username" type="text"
name="username">
```

Incluir alternativas para imágenes

Para situaciones en las cuales el usuario no aprecia bien las imágenes o bien no se cargan correctamente, conviene incluir un texto alternativo con el atributo “alt” en la etiqueta o también <picture>. Si la imagen no es parte del contenido y sólo es decorativa, es posible que no sea necesario incluir texto alternativo, pero es esencial para imágenes del contenido principal.

Identificar el lenguaje de la página y configuración

Identificar el lenguaje de la página en la etiqueta <html> es muy importante para que las extensiones y herramientas de traducción funcionen correctamente. No sólo se trata de traducir a idiomas comunes, sino que también es importante para la traducción al braille para discapacitados visuales. También permite facilitar las mejoras en pronunciación de los reproductores de voz, así como extensiones de diccionario cuando se selecciona el texto. El siguiente ejemplo señala el francés como el idioma de la página:

```
<!DOCTYPE>
<html lang="fr">
<head>
  <title>document écrit en français</title>
  <meta http-equiv="content-type" content="text/html; charset=utf-8" />
</head>
<body>
  ...document écrit en français...
</body>
</html>
```

Etiquetas semánticas y roles

Las etiquetas de html semántico son muy útiles para segmentar y estructurar el código correctamente, pero también tienen un papel de accesibilidad. El navegador las identifica y les atribuye automáticamente su rol. Las etiquetas semánticas son <header>, <nav>, <main>, <aside>, <section> y <footer>.

</> Code Snippet

```
<section>
  <article>
    <h2>Superbear saves the day</h2>
    <time datetime="2015-08-07">7 Aug
2015</time>
    <p>The city's favorite bear yet again
proves his mettle by rescuing a young cat
from a tree. Witnesses say that
Superbear's efforts were not appreciated
by the feline, who inflicted some minor
scratch wounds on his rescuer.</p>
    <aside>
      <h3>Related Articles</h3>
      <ul>
        <li><a href="#">Bear receives key
to city</a></li>
        <li><a href="#">Superbear stands
for mayor</a></li>
      </ul>
    </aside>
  </article>
</section>
```

Adicionalmente, conviene incluir roles para el resto de etiquetas cuando sea necesario a través del atributo “role”.

</> Code Snippet

```
<form action="#" method="post">
  <div role="search">
    <label for="search">Search
for</label>
    <input type="search" id="search"
aria-describedby="search-help">
    <div id="search-help">Search records
by customer id or name</div>
    <button type="submit">Go</button>
  </div>
</form>
```

Ayudar a los usuarios a evitar y corregir errores

Mostrar las razones por las que un campo de un formulario no se ha validado es muy útil para la experiencia de usuario. Conviene incluir instrucciones, ejemplos, razones del error, requisitos, etc:

Rendered

Phone

For example, (02) 1234 1234

</> Code Snippet

```
<label for="phone">Phone</label>
<input id="phone" name="phone" type="tel"
  pattern="^\(?\0[1-9]{1}\)?\?[0-9 -]*$"
  aria-describedby="phone-desc">
<p id="phone-desc">For example, (02) 1234 1234</p>
```


Reflejar el orden de lectura en el orden del código

Es importante que la estructura del código sea similar al orden de lectura de los elementos, independientemente de que se pueda lograr la misma visualización con orden diferente de etiquetas. En el siguiente ejemplo, el título va a antes de la imagen en el segundo trozo de código, mientras que en el primer snippet la imagen va antes:



Space trainers


Space trainer for a classic and stylish look.

 Add to cart

✖ Image before heading may be missed

```


<h3>Space trainers</h3>
<p>Space...</p>
<a href="...">Add to cart</a>
```

 View complete code example

✔ Heading marks the start of the section

```
<h3>Space trainers</h3>

<p>Space...</p>
<a href="...">Add to cart</a>
```

 View complete code example

Código que se adapta a la tecnología del usuario

La maquetación y estructura del contenido debe adaptarse a la pantalla del dispositivo. Aquí se pueden utilizar herramientas como flexbox o las media queries que reorganizan el contenido a medida que se reduce el ancho de pantalla o cambia el tipo de puntero.

Proveer significado a los elementos interactivos no estándar

WAI-ARIA permite definir y estructurar los elementos de la interfaz para herramientas de accesibilidad, por ejemplo asignando funciones o roles a determinados elementos. Es especialmente importante añadir los atributos correspondientes a los elementos que por defecto no tienen un rol asignado, como por ejemplo:

Example: Menu function and state identified using WAI-ARIA

```
<nav aria-label="Main Navigation" role="navigation">
  <ul>
    <li><a href="...">Home</a></li>
    <li><a href="...">Shop</a></li>
    <li class="has-submenu">
      <a aria-expanded="false" aria-haspopup="true"
href="...">SpaceBears</a>
      <ul>
        <li><a href="...">SpaceBear 6</a></li>
        <li><a href="...">SpaceBear 6 Plus</a></li>
      </ul>
    </li>
    <li><a href="...">MarsCars</a></li>
    <li><a href="...">Contact</a></li>
  </ul>
</nav>
```


Asegurar que todos los elementos interactivos son accesibles desde el teclado

</> Code Snippet

```
var buttonExample =
document.getElementById('example-button');

buttonExample.addEventListener('keydown',
function(e) {
    // Toggle the menu when RETURN is
    pressed
    if(e.keyCode && e.keyCode == 13) {

toggleMenu(document.getElementById('example-
button-menu'));
    }
});

buttonExample.addEventListener('click',
function(e) {
    // Toggle the menu on mouse click

toggleMenu(document.getElementById('example-
button-menu'));
});
```

Evitar códigos CAPTCHA siempre que sea posible

Los códigos CAPTCHA generan problemas para personas con problemas de visión. Por eso es importante que sean simples, o bien, que haya alternativas para personas discapacitadas.

3. HERRAMIENTAS DE DISEÑO

Las herramientas de diseño consisten en el uso de patrones de diseño que mejoren la accesibilidad, independientemente de cómo se implementen por parte del desarrollador.

Proveer suficiente contraste entre el plano principal y el secundario

Es importante que el contenido tenga suficiente contraste con el color de fondo. Una tipografía de color oscuro debería ir con un color de fondo claro y viceversa.

Utilizar diferenciaciones no basadas sólo en el color

Las personas con problemas de visión no pueden diferenciar contenidos sólo en base al color, por lo que es interesante aportar otras marcas adicionales como en el siguiente ejemplo:

Example: Refer to something using color alone

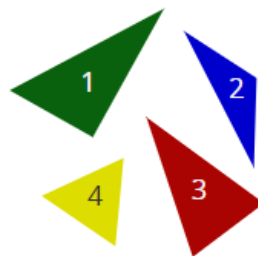
✖ Color only



Which is the right-angled triangle?

- ☐ Green
- ☐ Blue
- ☐ Red
- ☐ Yellow
- ☐ Don't know

✔ Color and number



Which is the right-angled triangle?

- ☐ Green (1)
- ☐ Blue (2)
- ☐ Red (3)
- ☐ Yellow (4)
- ☐ Don't know

Asegurar que los elementos interactivos son fáciles de identificar

Añadir marcas o colores ayuda a identificar aún más los elementos interactivos, facilitando la navegación a personas con problemas de visión. Concretamente,

se recomienda aportar estilos a los enlaces, tanto en el color de la fuente como en el color de fondo.

✔ Mouse hover style

[keyboard to navigate](#)



✔ Keyboard focus style

[keyboard to navigate](#)

✔ Touch or click style

[keyboard to navigate](#)



Proveer opciones de navegación claras y consistentes

Es importante que los nombres de los recursos en la navegación tengan nombres consistentes, así como aportar formas alternativas de navegación como buscadores o site maps.

Aportar feedback fácilmente identificable

Señalar con colores, iconos y marcas los avisos y errores en un formulario es un ejemplo de feedback que mejora la accesibilidad de la web.

Utilizar encabezados y espacios para el contenido agrupado

Utilizar encabezados y un espaciado correcto ayuda a identificar y agrupar el contenido de forma visual, mejorando la accesibilidad.

Incluir alternativas para imágenes y otros contenidos de media en el diseño

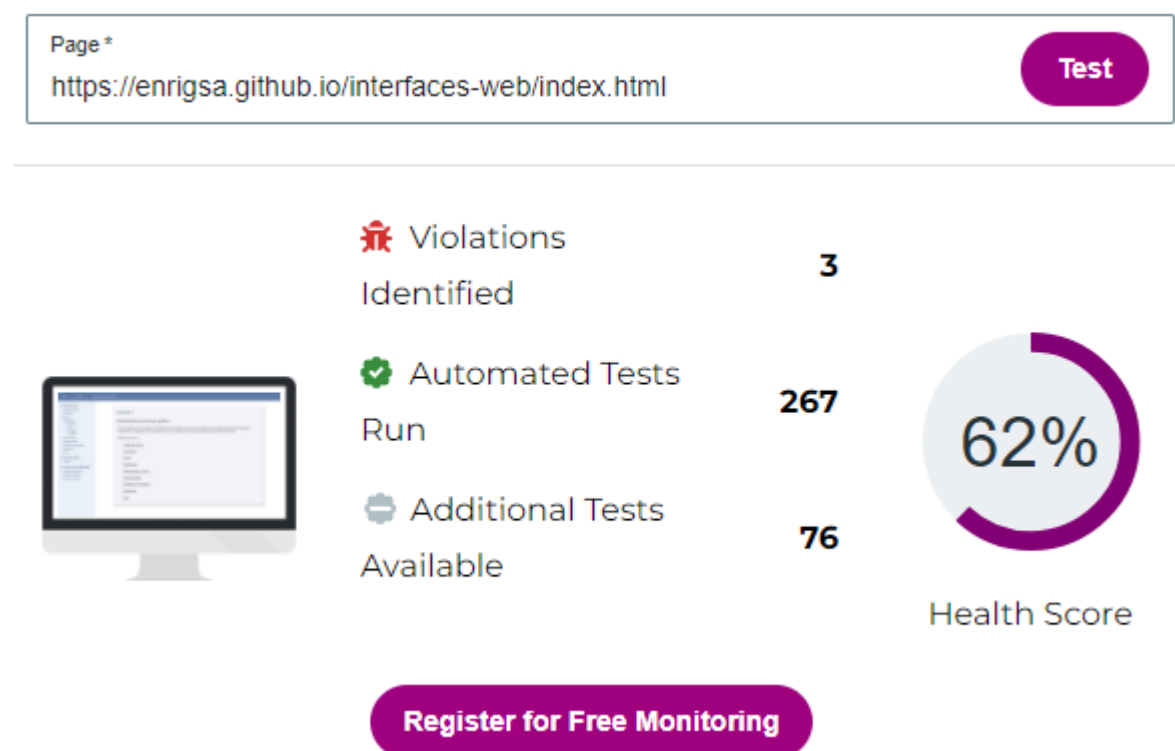
Aportar una transcripción es un ejemplo de mejora de accesibilidad para personas que no pueden escuchar un vídeo, ya sea por discapacidad o por circunstancias especiales.

Aportar controles para contenido que se ejecuta automáticamente

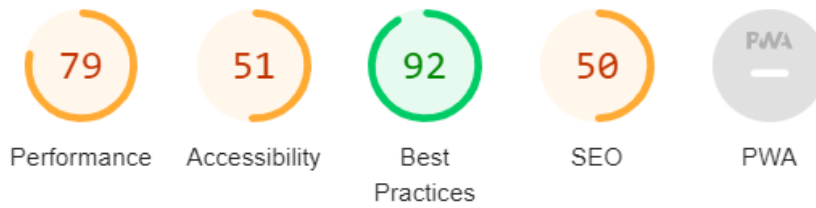
Los controles para controlar el contenido son importantes para contenido que se reproduce automáticamente, ya que puede ser necesario parar de forma inmediata su reproducción.

4. HERRAMIENTAS DE AUDITORÍA

La web de W3G tiene un apartado donde se muestran cientos de herramientas de auditoría de accesibilidad para webs. A continuación, mostramos una de las más cómodas de usar: [A-Tester de Evaluera Ltd.](#) Realizamos una auditoría de nuestro sitio web en Github Pages:



También realizamos un test con la herramienta de Google Lighthouse, que va más allá de cuestiones de accesibilidad pero que también indica aspectos de ésta:



There were issues affecting this run of Lighthouse:

- Chrome extensions negatively affected this page's load performance. Try auditing the page in incognito mode or from a Chrome profile without extensions.



Performance

Values are estimated and may vary. The [performance score is calculated](#) directly from these metrics. [See calculator](#).

▲ 0–49 ■ 50–89 ● 90–100



DIAGNOSTICS

- ▲ Image elements do not have explicit `width` and `height` ▼
- ▲ Does not have a `<meta name="viewport">` tag with `width` or `initial-scale` No `<meta name="viewport">` tag found ▼
- Minimize main-thread work — 2.6 s ▼
- Avoid chaining critical requests — 4 chains found ▼
- Keep request counts low and transfer sizes small — 7 requests • 12 KiB ▼
- Largest Contentful Paint element — 1 element found ▼
- Avoid long main-thread tasks — 3 long tasks found ▼

More information about the performance of your application. These numbers don't [directly affect](#) the Performance score.

Se pueden encontrar múltiples herramientas de tests y auditorías en el [apartado correspondiente de W3G](#).

5. Fuentes y recursos

- [Designing for Web Accessibility](#)
- [Developing for Web Accessibility](#)
- [Web Accessibility Evaluation Tools List](#)
- [A-Tester - Web Accessibility](#)