



Conceptos de Accesibilidad

Diseño de interfaces
Web

ESTHER CID

2º DAW



ÍNDICE

Desarrollando para Accesibilidad Web	3
Asociar una etiqueta con cada control de formulario	3
Incluir texto alternativo para las imágenes.....	3
Identificar el idioma de la página y los cambios de idioma	3
Utilice el marcado para transmitir significado y estructura.....	3
Ayudar a los usuarios a evitar y corregir errores	4
Refleja el orden de lectura en el orden del código.	5
Escribir código que se adapte a la tecnología del usuario	6
Proporcionar significado para elementos interactivos no estándar.....	6
Asegúrese de que todos los elementos interactivos sean accesibles desde el teclado	7
Evite CAPTCHA siempre que sea posible	8

Desarrollando para Accesibilidad Web

En este documento se presentarán ciertas consideraciones para tener en cuenta a la hora de desarrollar una web más accesible.

Estos consejos son buenas prácticas para ayudarlo a cumplir con los requisitos de las Pautas de accesibilidad al contenido web (WCAG).

Asociar una etiqueta con cada control de formulario

Use un `for` atributo en el `<label>` elemento vinculado al `id` atributo del elemento de formulario, o use atributos WAI-ARIA . En situaciones específicas, puede ser aceptable ocultar `<label>` elementos visualmente, pero en la mayoría de los casos se necesitan etiquetas para ayudar a todos los lectores a comprender la entrada requerida.

Ejemplo: Uso <code>for</code> y <code>id</code> atributos	
 prestado	</>Fragmento de código
Nombre de usuario <input type="text"/>	<pre><label for="username">Username</label> <input id="username" type="text" name="username"></pre>

Incluir texto alternativo para las imágenes

Use texto alternativo vacío, `alt=""` para las imágenes, o inclúyelos en el CSS en su lugar. Las alternativas de texto suelen ser proporcionadas por los responsables del contenido escrito.

Identificar el idioma de la página y los cambios de idioma

Indicar el idioma principal de cada página mediante el `lang` atributo en la `html` etiqueta, por ejemplo `<html lang="en">`.

Utilice el marcado para transmitir significado y estructura

Use el marcado apropiado para encabezados, listas, tablas, etc. HTML5 proporciona elementos adicionales, como `<nav>` y `<aside>`, para estructurar mejor su contenido.

Utilizar `role="search"` para identificar la funcionalidad de búsqueda.

Ejemplo: uso de HTML para proporcionar estructura y significado prestado**Superoso salva el día**

7 de agosto de 2015

El oso favorito de la ciudad demuestra una vez más su valor al rescatar a un gato joven de un árbol. Los testigos dicen que los esfuerzos de Superbear no fueron apreciados por el felino, quien infligió algunos rasguños menores a su salvador.

Artículos relacionados

- [Bear recibe la llave de la ciudad](#)
- [Superbear se postula para alcalde](#)

</>Fragmento de código

```
<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>
```

Ejemplo: campo de búsqueda usando WAI-ARIA prestado

Buscar

Buscar registros por ID o nombre de cliente

Vamos

</>Fragmento de código

```
<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

Proporcione instrucciones claras, mensajes de error y notificaciones para ayudar a los usuarios a completar formularios en su sitio. Cuando ocurre un error:

- Ayudar a los usuarios a encontrar dónde está el problema
- Proporcionar explicaciones específicas y comprensibles.
- Sugerir correcciones

Ejemplo: campo de número de teléfono australiano con validación indulgente

prestado

Teléfono

Por ejemplo, (02) 1234 1234

</>Fragmento de código

```
<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>
```

Refleja el orden de lectura en el orden del código.

El orden en el que aparezcan los elementos en el código deben coincidir con el orden cronológico de la información que aparece.

Una forma en la que podemos verificarlo es eliminar el estilo CSS y revisar que el orden del contenido tenga sentido.

Ejemplo: Reflejar el orden lógico de lectura en el código



Entrenadores espaciales

Entrenador espacial para un look clásico y elegante.

[Añadir a la cesta](#)

⚠ Es posible que se pierda la imagen antes del título

```

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

[+ Ver ejemplo de código completo](#)

✔ El encabezado marca el inicio de la sección.

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

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

[+ Ver ejemplo de código completo](#)

Escribir código que se adapte a la tecnología del usuario

Utilice un diseño receptivo para adaptar la pantalla a diferentes estados de zoom y tamaños de ventana gráfica. Cuando el tamaño de fuente aumenta al menos en un 200 %, evite el desplazamiento horizontal y evite cualquier recorte de contenido. Utilice la mejora progresiva para ayudar a garantizar que la funcionalidad y el contenido principal estén disponibles independientemente de la tecnología que se utilice.

Ejemplo: uso de consultas de medios para adaptar la navegación

```
/* On narrow viewports, make the navigation full width */
@media screen and (min-width: 25em) {
  #nav {
    float: none;
    width: auto;
  }
  #main {
    margin-left: 0;
  }
}
/* On wider viewports, put the navigation on the left */
@media screen and (min-width: 43em) {
  #nav {
    float: left;
    width: 24%;
  }
  #main {
    margin-left: 27%;
  }
}
```

Proporcionar significado para elementos interactivos no estándar.

Utilice WAI-ARIA para proporcionar información sobre la función y el estado de los widgets personalizados, como acordeones y botones personalizados.

Por ejemplo, `role="navigation"` y `aria-expanded="true"`. Se requiere código adicional para implementar el comportamiento de dichos widgets, como expandir y contraer contenido o cómo responde el widget a los eventos del teclado.

Ejemplo: función y estado del menú identificados mediante 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>
```

Asegúrese de que todos los elementos interactivos sean accesibles desde el teclado

Piense en el acceso al teclado, especialmente cuando desarrolle elementos interactivos, como menús, información sobre el mouseover, acordeones plegables o reproductores multimedia.

Úselo `tabindex="0"` para agregar un elemento que normalmente no recibe el foco, como `<div>` o ``, en el orden de navegación cuando se usa para la interacción. Use secuencias de comandos para capturar y responder a los eventos del teclado.

Ejemplo: botón de menú accesible desde el teclado

prestado

≡ Menú

</Fragmento de código

```
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'));
});
```


Evite CAPTCHA siempre que sea posible

Existen otros medios para verificar que la entrada del usuario fue generada por un ser humano que son más fáciles de usar, como la detección automática o las interacciones de la interfaz.

Si realmente necesita incluir CAPTCHA, debe seguir las siguientes pautas para que sea lo más accesible posible:

- Proporcionando más de dos formas de resolver los CAPTCHA
- Proporcionar acceso a un representante humano que puede omitir CAPTCHA
- No requiere CAPTCHA para usuarios autorizados.

IMPLEMENTACIÓN EN ALGUNAS PÁGINAS QUE YA HEMOS CREADO

El primer ejemplo lo desempeñamos en un formulario que creamos específicamente para probar la accesibilidad.

Vemos como está asociado a etiquetas <label>.

```
<form action="#">
  <label for="nombre">Nombre:</label><br>
  <input type="text" id="nombre" name="nombre" class="redondeo cuadro"><br>
  <label for="tef">Teléfono:</label><br>
  <input type="number" id="tef" name="tef" class="redondeo cuadro"><br><br>
  <input type="submit" value="Enviar" class="redondeo boton">
  <input type="reset" value="Limpiar" class="redondeo boton">
</form>
```

Para mejorar la accesibilidad en las imágenes lo que tenemos que hacer es incluir un texto alternativo que especifique que hay en ellas.

```
<div>
  <article>
    <!--La idea es colocar las imágenes utilizando solo HTML y controlando los tamaños con HTML-->
    <!--He cortado las imágenes en tres tamaños diferentes 800w, 480w, 320w
    Colocando de forma predeterminada la imagen de 800w-->
    
  </article>
  <article>
    
  </article>
</div>
<div>
  <article>
    
  </article>
  <article>
    
  </article>
</div>
```


Debemos especificar el idioma en el que se encuentra la página que vamos a desarrollar.

```

<!DOCTYPE html>
<html lang="es">

```

Debemos utilizar las etiquetas semánticas que nos proporciona HTML5 para que quede clara la parte de la página en la que se encuentra cada elemento.

Podemos utilizar el atributo “role” para ayudar a la comprensión de la utilización de cada etiqueta.

```

22. Conceptos_Accesibilidad > index.html > html
1  <!DOCTYPE html>
2  <html lang="es">
3  <head>
4      <meta charset="UTF-8">
5      <meta http-equiv="X-UA-Compatible" content="IE=edge">
6      <meta name="viewport" content="width=device-width, initial-scale=1.0">
7      <link rel="stylesheet" href="css/estilos.css">
8      <title>Formulario - Accesibilidad</title>
9  </head>
10 <body>
11     <header role="banner">
12         <h1>Prueba de accesibilidad con formularios</h1>
13     </header>
14     <main role="elements">
15         <section role="partition">
16             <article class="redondeo" role="article">
17                 <form action="#" role="formulario">
18                     <label for="nombre">Nombre:</label><br>
19                     <input type="text" id="nombre" name="nombre" class="redondeo cuadro"><br>
20                     <label for="tef">Teléfono:</label><br>
21                     <input type="number" id="tef" name="tef" class="redondeo cuadro"><br><br>
22                     <input type="submit" value="Enviar" class="redondeo boton">
23                     <input type="reset" value="Limpiar" class="redondeo boton">
24                 </form>
25             </article>
26         </section>
27     </main>
28     <footer class="footer" role="pie">
29         Página creada por Esther Cid
30     </footer>
31 </body>
32 </html>

```

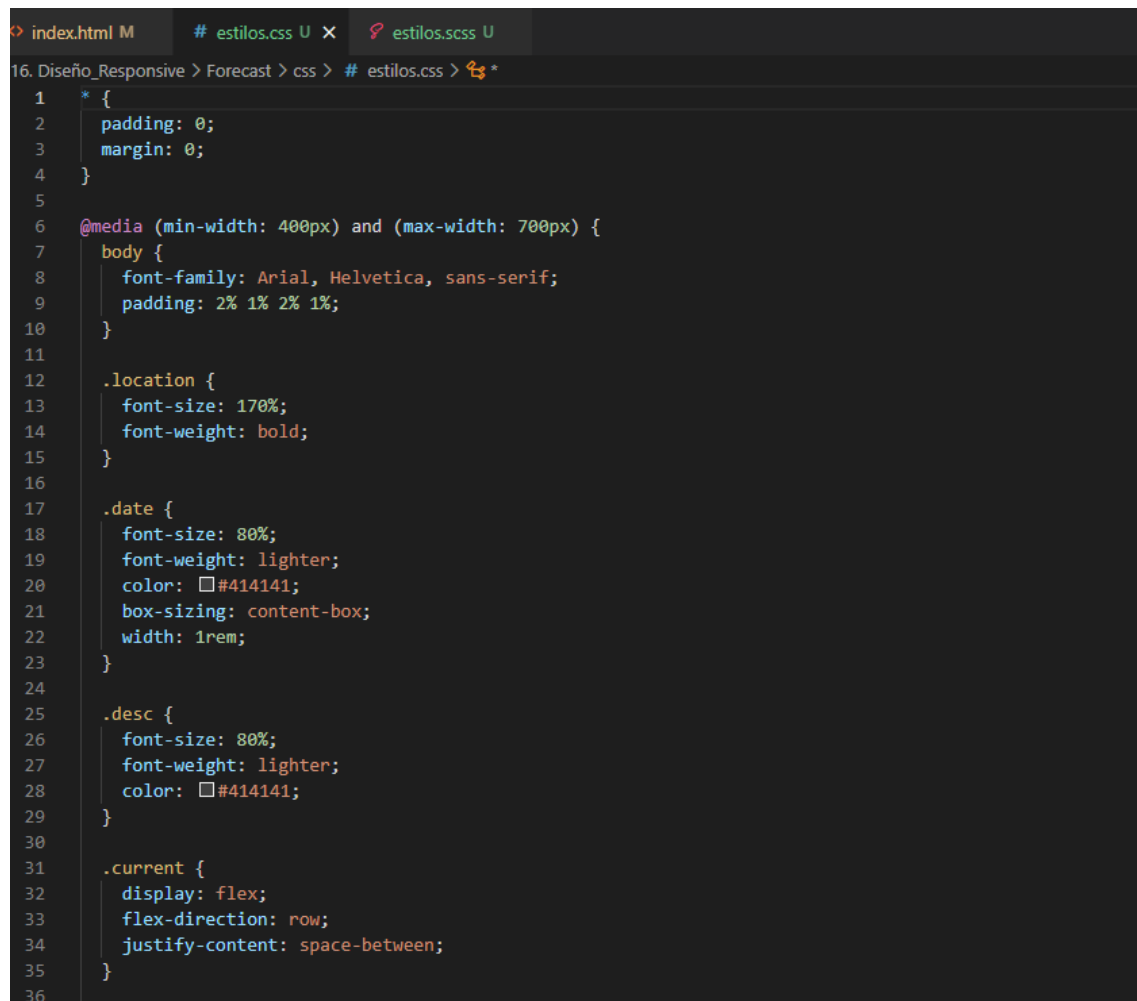
Los formularios pueden tener un patrón que limite el envío de datos que no se ajusten a él. Otra de las cosas que tenemos que hacer es ayudar al usuario señalándole donde se encuentra el error.

```

<input type="number" id="tef" name="tef" pattern="^(?0[1-9]{1}){1}?)?[0-9-]*$" class="redondeo cuadro"><br><br>

```

El código debe adaptar el contenido en función de la pantalla en la que se vea, de esta forma lo podemos hacer más accesible en caso de que se acceda mediante un dispositivo con la pantalla más pequeña como una Tablet o un teléfono móvil.



```
1  * {
2    padding: 0;
3    margin: 0;
4  }
5
6  @media (min-width: 400px) and (max-width: 700px) {
7    body {
8      font-family: Arial, Helvetica, sans-serif;
9      padding: 2% 1% 2% 1%;
10   }
11
12   .location {
13     font-size: 170%;
14     font-weight: bold;
15   }
16
17   .date {
18     font-size: 80%;
19     font-weight: lighter;
20     color: #414141;
21     box-sizing: content-box;
22     width: 1rem;
23   }
24
25   .desc {
26     font-size: 80%;
27     font-weight: lighter;
28     color: #414141;
29   }
30
31   .current {
32     display: flex;
33     flex-direction: row;
34     justify-content: space-between;
35   }
36 }
```

```
82  @media (min-width: 700px) and (max-width: 1200px) {
83    body {
84      font-family: Arial, Helvetica, sans-serif;
85      padding: 2% 1% 2% 1%;
86    }
87
88    .location {
89      font-size: 170%;
90      font-weight: bold;
91    }
92
93    .date {
94      font-size: 80%;
95      font-weight: lighter;
96      color: #414141;
97      box-sizing: content-box;
98      width: 1rem;
99    }
100
101    .desc {
102      font-size: 80%;
103      font-weight: lighter;
104      color: #414141;
105    }
106
107    .current {
108      display: flex;
109      flex-direction: row;
110      justify-content: space-between;
111    }
112  }
```

También podemos utilizar una serie de etiquetas que son capaces de, utilizando una serie de widgets personalizados, indicar si ese elemento transmite el estado actual del componente o si, por ejemplo, el botón funciona como una casilla de verificación.

```
1  <!DOCTYPE html>
2  <html lang="en">
3    <head>
4      <meta charset="UTF-8" />
5      <meta http-equiv="X-UA-Compatible" content="IE=edge" />
6      <meta name="viewport" content="width=device-width, initial-scale=1">
7      <link rel="stylesheet" href="css/estilos.css" />
8      <title>DIW-Esther Cid</title>
9    </head>
10   <body>
11     <!--Este es el menú del contenedor-->
12     <div id="entrada" class="redondeo">
13       <div id="cuerpo">
14         <div id="descripcion" class="centrar">
15           <div id="presentacion" aria-expanded="true">
16             <h1>ESTHER CID</h1>
17             <h4>FULLSTACK DEVELOPER</h4>
18             <p>Lorem ipsum</p>
19           </div>
20         </div>
21         <div id="dibujo" class="centrar">
22           
23         </div>
24       </div>
25       <div id="navegador" role="selector" aria-expanded="true">
26         <h1 class="boton" aria-controls="navegador">PRIMER TRIMESTRE</h1>
27         <h1 class="boton" aria-controls="navegador">SEGUNDO TRIMESTRE</h1>
28       </div>
29     </div>
30   </body>
31 </html>
32
```

PRUEBAS ACCESIBILIDAD CON WAVE

The screenshot displays the WAVE web accessibility evaluation tool interface. At the top, the WAVE logo is shown next to the text "powered by WebAIM" and "web accessibility evaluation tool". Below this, a toggle switch for "Styles" is set to "ON". The main section is titled "Summary" and features a navigation bar with icons for Summary, Details, Reference, Structure, and Contrast. The Summary view shows a grid of six metrics: Errors (0), Contrast Errors (0), Alerts (0), Features (5), Structural Elements (4), and ARIA (18). A blue button labeled "View details" is positioned below the grid. At the bottom, a congratulatory message states: "Congratulations! No errors were detected! Manual testing is still necessary to ensure compliance and optimal accessibility."

WAVE
web accessibility evaluation tool

powered by
[WebAIM](#)

Styles: OFF ☒ ON

Summary

Summary Details Reference Structure Contrast

0 Errors	0 Contrast Errors
0 Alerts	5 Features
4 Structural Elements	18 ARIA

View details >

Congratulations! No errors were detected! Manual testing is still necessary to ensure compliance and optimal accessibility.

