



# Animaciones y accesibilidad





CSS problems



# Animando con CSS

Las animaciones con css son... un quilombo. Hay un montón de propiedades que podemos aplicar, así que vamos a ir de a poco.

- `@keyframes`
  - `animation-name`
  - `animation-duration`
  - `animation-delay`
  - `animation-iteration-count`
  - `animation-direction`
  - `animation-timing-function`
  - `animation-fill-mode`
  - `animation`
- 



Los que vengan del palo visual probablemente reconozcan los keyframes; básicamente podríamos considerar que es el qué, cuándo, y la forma más sencilla de ellos sería esta, de **from** a **to**, si no nos preocupa exactamente en qué porcentaje de la animación va a estar pasando:

```
div {  
  width: 100px;  
  height: 100px;  
  background-color: red;  
  animation-name: example;  
  animation-duration: 4s;  
}  
  
@keyframes example {  
  from {background-color: red;}  
  to {background-color: yellow;}  
}
```



Podemos poner porcentajes, y con eso ajustar un poco mejor exactamente cuándo pasa cada cosa. En este caso, cada keyframe va a durar un segundo, porque la animación es de 4:

```
/* The animation code */
@keyframes example {
  0%   {background-color: red;}
  25%  {background-color: yellow;}
  50%  {background-color: blue;}
  100% {background-color: green;}
}

/* The element to apply the animation to */
div {
  width: 100px;
  height: 100px;
  background-color: red;
  animation-name: example;
  animation-duration: 4s;
}
```



Se puede combinar qué propiedades van a cambiar con cada keyframe, no solo poner una:

```
/* The animation code */
@keyframes example {
  0%   {background-color:red; left:0px; top:0px;}
  25%  {background-color:yellow; left:200px; top:0px;}
  50%  {background-color:blue; left:200px; top:200px;}
  75%  {background-color:green; left:0px; top:200px;}
  100% {background-color:red; left:0px; top:0px;}
}

/* The element to apply the animation to */
div {
  width: 100px;
  height: 100px;
  position: relative;
  background-color: red;
  animation-name: example;
  animation-duration: 4s;
}
```



Se puede poner un retardo con animation-delay:

```
div {  
  width: 100px;  
  height: 100px;  
  position: relative;  
  background-color: red;  
  animation-name: example;  
  animation-duration: 4s;  
  animation-delay: 2s;  
}
```

Y decidir cuántas veces corre con animation-iteration-count. Para que no frene nunca el valor es infinite:

```
div {  
  width: 100px;  
  height: 100px;  
  position: relative;  
  background-color: red;  
  animation-name: example;  
  animation-duration: 4s;  
  animation-iteration-count: 3;  
}
```



Se pueden combinar todos en una sola propiedad, animation:

```
div {  
  animation-name: example;  
  animation-duration: 5s;  
  animation-timing-function: linear;  
  animation-delay: 2s;  
  animation-iteration-count: infinite;  
  animation-direction: alternate;  
}
```



```
div {  
  animation: example 5s linear 2s infinite alternate;  
}
```





**ANIMATE**



**ALL THE THINGS**





Coding.area.51

@Codingarea51

When you forget to attach Css file





Let's see who caused this bug.



## Les dejo esto para jugar:

- Pueden probar esto con la hamburguesa del menú...

<https://codepen.io/chriswrightdesign/pen/fAayG>

- Esto con... lo que quieran, es divertido:

<https://codepen.io/darkwing/pen/bCali>

- guárdense esto, cuando empecemos a ver spinners les va a venir bien:

<https://codepen.io/jenning/pen/YzNmzaV>

- para botones:

<https://codepen.io/4views/pen/qwJxop>

<https://codepen.io/four-leaf-4/pen/XVqYgN>

<https://codepen.io/WE4/pen/KKwEZre>

<https://codepen.io/Aisik28/pen/XPoQez>



# Accesibilidad:

La accesibilidad no es un tema menor, aunque muchas veces se la considera a último momento y se aplica poco y mal. Hacer un sitio accesible es hacer que un sitio sea **para todos**.

Algo que no debemos olvidar es: hacer un sitio accesible a uno, lo hace mejor para todos, y el 90% de las personas vamos a tener algún tipo de discapacidad en algún momento de nuestra vida. A qué me refiero? no ves nada sin lentes? discapacidad visual, literalmente los anteojos o lentes de contacto son como las muletas. Te rompiste un brazo? seguro ahí el mouse te cuesta más usarlo. Estado de nervios extremo? probablemente nuestra mente va a estar nublada y no vamos a poder interpretar correctamente el sitio (eso se resuelve con el **diseño de crisis**).

Accesible para uno es **accesible para todos**, y jamás me voy a cansar de insistir sobre eso.

Pero solo quien lo vive sabe qué es lo que hace que algo le sea accesible.



La forma más básica y sencilla de accesibilidad la encontramos usando html semántico, porque le da contexto a los lectores de pantalla

## Semántico

```
<button>Click Me</button>
```

## No semántico

```
<div>Click Me</div>
```














Acá les dejo el apartado de accesibilidad de mozilla:

[https://developer.mozilla.org/es/docs/Learn/Accessibility/What\\_is\\_accessibility](https://developer.mozilla.org/es/docs/Learn/Accessibility/What_is_accessibility)

y el de w3s: [https://www.w3schools.com/html/html\\_accessibility.asp](https://www.w3schools.com/html/html_accessibility.asp)


## Personas con discapacidad visual

Las personas con discapacidad visual son aquellas con ceguera, poca visión o daltonismo. Muchas personas con discapacidad visual utilizan amplificadores de pantalla que consisten en lupas físicas o funciones de zoom por software. La mayoría de los navegadores y sistemas operativos de hoy en día tienen instaladas funciones de zoom. Algunos usuarios confían en los lectores de pantalla, un software que lee en voz alta los textos digitales. Algunos ejemplos de lectores de pantalla incluyen:

- Productos comerciales de pago, como [JAWS](#)  (Windows) y [Dolphin Screen Reader](#)  (Windows).
- Productos gratuitos, como [NVDA](#)  (Windows), [ChromeVox](#)  (Chrome, Windows y Mac OS X) y [Orca](#)  (Linux).
- Productos integrados en el sistema operativo, como [VoiceOver](#)  (MacOS, iPadOS, IOS), [Narrator](#)  (Microsoft Windows), [ChromeVox](#)  (en Chrome OS) y [TalkBack](#)  (Android).



## Personas con discapacidades auditivas

También conocidas como personas con trastornos auditivos o personas sordas. Son un grupo de personas con niveles de audición bajos o nulos. Las personas con discapacidad auditiva usan AT (véase [Dispositivos de asistencia para personas con trastornos auditivos, de voz, del habla o del lenguaje](#) ) , pero en realidad no hay AT específicos para el uso del ordenador/web.


Hay, sin embargo, técnicas específicas para ofrecer alternativas textuales a contenidos de audio, que van desde simples transcripciones hasta pistas de texto (es decir, subtítulos) que se pueden mostrar junto con el vídeo. Pero lo veremos más adelante.

Personalmente, considero que si tu sitio se apoya 100% en un video narrado, o tiene un video que vende el producto, ponerle subtítulos es indispensables. No puede ser que una persona pierda parte de contenido si no puede acceder a la narración.





## Personas con discapacidad motriz

Estas personas tienen discapacidades relativas a la movilidad, que pueden implicar problemas puramente físicos (como la pérdida de una extremidad o la parálisis) o trastornos neurológicos/genéticos que conducen a la debilidad o pérdida de control en las extremidades. Algunas personas simplemente pueden tener dificultades a la hora de mover el ratón, mientras que otras podrían verse más gravemente afectadas, tal vez estén paralizadas y necesiten utilizar un [puntero de cabeza](#)  para interactuar con los ordenadores.

Este tipo de discapacidad se da principalmente por la vejez, y no por cualquier trauma o condición específica, y también podría resultar de limitaciones en el hardware (algunos usuarios podrían no tener un ratón).

La forma en que esto afecta al desarrollo web es el requisito de que los controles sean accesibles por el teclado. Hablaremos de la accesibilidad del teclado en artículos posteriores de este módulo, pero te recomendamos probar algunos sitios web utilizando solo el teclado para ver cómo funcionan. Por ejemplo, ¿se puede utilizar la tecla de tabulación para moverse entre los diferentes controles de un formulario web? Puedes encontrar más detalles sobre los controles del teclado en nuestro apartado [Accesibilidad desde el teclado](#).

Todo sitio tiene que ser 100% navegable con tab. Punto.



## Personas con discapacidad cognitiva

La discapacidad cognitiva engloba una amplia gama de discapacidades, desde las personas que presentan las capacidades intelectuales más limitadas hasta toda la población que tiene problemas a la hora de recordar por los síntomas de la edad. Este amplio abanico incluye a las personas con enfermedades mentales como la [depresión](#) y la [esquizofrenia](#). También incluye a personas con dificultades de aprendizaje, como la [dislexia](#) y el [trastorno por déficit de atención con hiperactividad](#). Es importante destacar que, aunque hay una gran diversidad dentro de las definiciones clínicas de alteraciones cognitivas, las personas que las experimentan tienen un conjunto común de problemas funcionales, que incluye dificultades a la hora de entender los contenidos, recordar cómo completar las tareas y confusión ante páginas web diseñadas de forma incoherente.

Una buena base de accesibilidad para personas con deficiencias cognitivas incluye:

- proporcionar el contenido en más de un formato, como puede ser texto-a-voz o vídeo;
- proporcionar contenidos fáciles de entender, como texto escrito con estándares de lenguaje sencillo;
- centrar la atención en el contenido importante;
- minimizar las distracciones, tales como contenidos innecesarios o anuncios;
- proporcionar un diseño coherente de la página web y del sistema de navegación;
- usar elementos ya conocidos, como los enlaces subrayados en azul cuando aún no se han visitado, y en morado cuando sí;
- dividir los procesos en pasos lógicos y esenciales con indicadores de progreso;
- ofrecer un sistema de autenticación del sitio web de la forma más fácil posible sin comprometer la seguridad; y
- diseñar formularios fáciles de completar, con mensajes de error claros y de fácil solución.

Respecto de este punto tengo muchas opiniones, y no todas positivas. Nadie puede saber cómo alguien en un espectro percibe las cosas excepto que esté en él, así que cuando veo “casos de autistas usando sitios” o “simulador de dislexia”... sí, no les crean mucho. Estos consejos son correctos y adhiero a ellos, pero solo alguien que viva dentro de cierta variedad mental sabe cómo se siente. Y ni que decir que considerar discapacidad cognitiva muchas de estas cosas es insultante.



# Diseño de crisis

Es el concepto de diseñar sitios a los que uno vaya a acceder en estados de agitación mental de forma que sean fáciles de navegar.

Esto aplica especialmente a sitios de clínicas, líneas de auxilio, etc. Si me corté un dedo claramente no tengo ganas de scrollear hasta abajo, entrar a contacto y tratar de entender a dónde debería llamar, no?



# Bueno, ahora a lo interesante. Cómo aplicarlo.

Porque hasta ahora mucho blabla y deben estar re podridos.

Hay varias extensiones de chrome que son súper útiles:

- [wave](#): excelente y súper clara. Te dice directamente qué hacer.
- [siteImprove](#): no te deja pasar nada, pero *nada*.
- [spectrum](#): para probar el sitio para diferentes tipos de daltonismo.
- [SilkTide](#): simulador de diferentes dificultades.
- [axe DevTools](#): me lo han recomendado mucho, pero no lo usé.
- [helperBird](#): no es para testear, sino para ver cómo una extensión para dislexia ayuda con la visualización.



Ninguna herramienta va a reemplazar jamás la experiencia humana. Siempre que puedan, testeen su sitio con gente real. Y solo la persona dentro de un grupo puede decirte cómo se siente ser parte.



# Títulos

Los motores de búsqueda utilizan los títulos para indexar la estructura y el contenido de sus páginas web.

Los usuarios hojean sus páginas por sus títulos. Es importante utilizar títulos para mostrar la estructura del documento y las relaciones entre las diferentes secciones.

`<h1>` Los títulos deben usarse para los títulos principales, seguidos de los `<h2>` títulos, luego los menos importantes `<h3>`, y así sucesivamente.

**Nota:** Utilice títulos HTML solo para títulos. No uses títulos para hacer que el texto sea **GRANDE** o en **negrita**.



## Texto alt

No es solo para casos en que el navegador no muestre la imagen; es lo que un lector de pantalla leería. Si la imagen no le es relevante, la etiqueta alt podemos dejarla, pero vacía (de qué le sirve a alguien escuchar la palabra “logo”, por ejemplo?).

### Ejemplo

```

```



# Idioma

Nos sirve por un lado para que google no nos insista en si queremos traducir, pero también, para los motores de búsqueda y lectores de pantalla.

Siempre debe incluir el `lang` atributo dentro de la `<html>` etiqueta, para declarar el idioma de la página web. Esto está destinado a ayudar a los motores de búsqueda y navegadores.

El siguiente ejemplo especifica el inglés como idioma:

```
<!DOCTYPE html>
<html lang="en">
<body>

...

</body>
</html>
```





### #3 Usa el atributo 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.

Buena info sobre cuáles existen: <https://www.neoguias.com/accesibilidad-web/>

La realidad es que si tenemos semántico, no solemos necesitar role, pero si por algún motivo necesitamos que sean divs, ahí nos viene bien.



Vamos a tomar como ejemplo un elemento `nav` que podríamos definir tal y como ves a continuación:

```
<nav>
  <ul>
    <li><a href="/">Inicio</a></li>
    <li><a href="/noticias">Noticias</a></li>
    <li><a href="/contacta">Contacta</a></li>
  </ul>
</nav>
```

Si por algún motivo te vieses obligado a usar una etiqueta `div` en lugar de una etiqueta `nav`, tendrías 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>
```

En resumen; el atributo `role` se usa para dar significado a aquellos elementos de la interfaz que carecen del mismo.



## #4 Usa los atributos **aria**

Los atributos ARIA definen la semántica que puede ser aplicada a los diferentes elementos HTML. ARIA es un acrónimo de **Accessible Rich Internet Applications**, que significa *Aplicaciones Enriquecidas Accesibles de Internet*. A continuación vamos a ver algunos de estos atributos:

- **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">
  Sinopsis de la película
</div>
```



- **aria-hidden:** En ocasiones podrías querer que los lectores de pantalla ignoren algunos elementos que no contienen información relevante. Para ello puedes 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:

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

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

```

```





- **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:

```
<textareas id="descripcion" aria-multiple="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:

```

```

- **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>
```

- **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</li>
  <li id="tab-2">Tab2</li>
</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</button>
<button onclick="gestionarPanel('B');" role="tab" aria-selected="elemento-b">Selector</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-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**. Aquí tienes un ejemplo de uso:

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



- **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-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>
```





## #5 Usa el atributo `tabindex`

El atributo `tabindex` se usa para cambiar el orden mediante el cual los elementos mostrados en pantalla se seleccionan cuando se pulsa el tabulador. Por defecto, solamente los enlaces y los formularios pueden ser seleccionados de esta forma, pudiendo saltar rápidamente de un elemento a otro. Sin embargo, podrías querer que los usuarios puedan seleccionar otros elementos.

El atributo `tabindex` acepta un número como valor que se corresponde con el orden de selección. Si agregas `tabindex="0"` a un elemento, será posible seleccionarlo siguiendo el flujo de selección habitual:

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

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

```
<div tabindex="-1">  
  ...  
</div>
```



De todos estos, los que más he visto usar son:

- alt
- aria-label
- aria-hidden
- aria-describedby
- aria-checked
- aria-disabled
- aria-expanded
- aria-invalid
- aria-selected
- tabindex

Sí, son bocha, pero con cosas como wave enseguida ven dónde necesitan qué!





**NEW THINGS?**

**I'M TERRIFIED OF THAT.**

makeameme.org

