

# HTML5

## Introducción a HTML5

HTML son las siglas de **HyperText Markup Language** o Lenguaje de Marcado de Hipertexto. Es un lenguaje basado en etiquetas.

El código HTML se escribe en un documento de texto con extensión `.html`. Este HTML es interpretado por un *"user agent"*, que en la mayoría de ocasiones se corresponde con un navegador web. Sin embargo existen otro tipo de *"user agent"* como los robots de indexación de los motores de búsqueda, etcétera...

El objetivo del HTML es describir la estructura del documento e indicar el **contenido semántico** de cada elemento que forma parte del documento.

Para ello se emplean las **etiquetas**, que son la base del lenguaje HTML. Existen muchas etiquetas y cada una se utiliza para contener información y darle un **significado semántico** a dicha información.

Esta es la razón por la que el lenguaje HTML se considera semántico. Describe el contenido de la información con el uso de las etiquetas. Para gestionar el aspecto visual de esa información se debe utilizar el lenguaje CSS.

El estándar HTML contiene un número concreto de etiquetas, algunas plenamente utilizables y otras marcadas como obsoletas. Sin embargo, aunque no se genere ningún error, en **HTML** no se debe utilizar cualquier palabra como etiqueta:

```
<!-- INCORRECTO -->
<etiqueta>contenido</etiqueta>

<!-- Correcto -->
<p>Párrafo</p>
```

En el documento HTML debe aparecer información correctamente individualizada, de modo que al leer una página HTML comprendamos su significado, y si queremos cambiar la apariencia, lo hagamos en el documento CSS. Esto es lo que comunmente se conoce como **separación de la presentación del contenido**.

Hola, quiero resaltar esta `<b>palabra</b>`.

En este caso se está utilizando la etiqueta `<b>` de HTML4 y anteriores para poner en negrita un texto. Se está utilizando una propiedad de presentación (visual) en el HTML, algo que no se debe hacer en HTML5. La misión de HTML5 es mantener sólo contenido e información semántica en HTML5. Por dicha razón, la forma de hacerlo en HTML5 es utilizando una etiqueta que añade esa información semántica como es `<strong>`:

Hola, quiero resaltar esta `<strong>palabra</strong>`.

Del mismo modo, en lugar de usar `<i>` para poner un texto en cursiva, en HTML5 se debería usar `<em>` para marcar énfasis:

Hola, quiero enfatizar esta `<em>palabra</em>`.

En los navegadores, hay varias formas de acceder al código HTML de la página:

- Pulsando la combinación de teclas `Ctrl + U` . Te aparecerá el código fuente tal cuál lo recibe el navegador.
- Pulsando `Ctrl + Shift + I` aparecerá la consola del navegador.

## Estructura de una etiqueta HTML

La estructura de las etiquetas HTML tiene el siguiente formato:

```
<etiqueta atributo="valor">contenido</etiqueta>
```

Por norma general, **las etiquetas deben cerrarse** para indicar donde finaliza su contenido.

Sin embargo, existen algunas etiquetas que no requieren un cierre explícito, como `<hr>` o `<img>` , ya que no contienen contenido textual o elementos anidados. Estas etiquetas se conocen como **etiquetas vacías o etiquetas autocontenidas**.

En versiones más antiguas de HTML, era común ver estas etiquetas con un cierre automático, como `<img />` , pero en HTML5 esto no es necesario.

## Formato de una etiqueta

La parte esencial de una etiqueta HTML es lo que se denomina la **etiqueta de apertura**.

Consiste en escribir el nombre de la etiqueta en cuestión, colocándolo entre los caracteres `<` y `>` . Aunque los navegadores pueden procesar etiquetas que no siguen este formato estrictamente, las etiquetas HTML deben escribirse siempre en **minúsculas** para cumplir con las buenas prácticas y con la especificación del estándar HTML.

Para que el código HTML sea válido, no se puede utilizar cualquier palabra como etiqueta, aunque el lenguaje HTML es permisivo en algunos aspectos y los navegadores puedan interpretar etiquetas no estándar.

La mayoría de las etiquetas requieren un **cierre de etiqueta** para indicar dónde termina su efecto. El cierre de la etiqueta se caracteriza por el uso del mismo nombre de la etiqueta de apertura, pero precedido por una barra `/` , inmediatamente después del `<` :

```
<p>Este es un párrafo</p>
```

## Atributos

En algunas etiquetas HTML, existen **atributos específicos** (que pueden ser **opcionales u obligatorios**) y otros que son **atributos globales**.

Los atributos proporcionan **información adicional** sobre una etiqueta y generalmente van acompañados de un **valor determinado**. Los atributos se colocan después del nombre de la etiqueta, separados por un espacio y antes del carácter `>` :

```
<strong id="dato">Contenido</strong>

<strong id="dato" class="clase1" lang="es">Contenido de texto</strong>
```

Una etiqueta puede tener uno o varios pares de **atributo-valor**, pero nunca se debe repetir el mismo atributo en una misma etiqueta, ya que el valor posterior **sobrecribiría** al anterior. El orden de los atributos no es importante.

Aunque los valores de los atributos pueden ir entre comillas simples o dobles, la convención es utilizar **comillas dobles** para mayor consistencia y legibilidad.

Existen 3 tipos principales de atributos según el tipo de valores que aceptan:

1. **Conjunto de valores:** atributos que sólo aceptan un conjunto predefinido de valores. Cualquier otro valor será inválido.
2. **Valores libres:** atributos donde puedes especificar un valor libremente, como una URL o texto, sin un conjunto predefinido de valores.
3. **Valores booleanos:** atributos que representan un valor verdadero o falso. En HTML5, si el atributo está presente sin un valor, se considera como verdadero ( `true` ), y si se omite, se interpreta como falso ( `false` ).

```
<!-- Ejemplo de atributo con un conjunto de valores válidos -->
<input type="email" placeholder="Introduce tu correo">

<!-- Ejemplo de atributo con valores libres -->
<a href="https://www.ejemplo.com">Visita nuestro sitio web</a>

<!-- Ejemplo de atributo booleano -->
<input type="checkbox" checked> Opción seleccionada por defecto
<br>
<button disabled>Botón deshabilitado</button>
```

## Contenido de la etiqueta

Una etiqueta HTML puede contener desde un simple fragmento de texto hasta un grupo de etiquetas anidadas. Este contenido puede incluir tanto texto plano como otras etiquetas HTML que estructuren o den formato al contenido:

```
<div id="pagina">
  <!-- Bloque de etiquetas anidadas -->
  <strong>Contenido importante</strong>
</div>
```

## Comentarios en HTML

Para introducir comentarios en el código HTML, se utilizan los delimitadores `<!-- y -->`. Todo lo que se encuentre entre estos delimitadores será considerado un comentario y no será visible en la página web, aunque sí estará presente en el código fuente.

```
<!-- Este es un comentario en HTML -->
<p>Este es un párrafo visible.</p>
```

 Los comentarios en HTML son **visibles** si se accede al código fuente del documento a través del navegador.

## Elementos en bloque vs elementos en línea

La distinción entre **elementos en bloque** y **elementos en línea** se utiliza en las especificaciones de HTML hasta la 4.01. HTML5 hereda esta noción, donde los elementos de estructura, como `<div>` o `<p>`, son elementos de bloque mientras que los elementos de formato de texto, como `<strong>` o `<span>`, son elementos en línea.

Un **elemento de bloque** ocupa todo el ancho de su elemento padre (el contenedor), creando un *"bloque"*. Los navegadores suelen mostrar los elementos de bloque con un salto de línea antes y después de ellos.

En términos de anidación, los elementos de tipo bloque pueden contener otros elementos de tipo bloque, elementos en línea y texto.

Por otro lado, un **elemento en línea** sólo ocupa el espacio delimitado por el contenido de sus etiquetas. De forma predeterminada, los elementos en línea no causan un salto de línea.

Los elementos en línea pueden contener otros elementos de tipo en línea y texto, pero no deben contener elementos de tipo bloque.

### Elementos de bloque

- `<address>`
- `<article>`
- `<aside>`
- `<audio>`
- `<blockquote>`
- `<canvas>`
- `<dd>`
- `<div>`
- `<dl>`
- `<fieldset>`
- `<figcaption>`
- `<figure>`
- `<footer>`
- `<form>`
- `<h1>`, `<h2>`, `<h3>`, `<h4>`, `<h5>`, `<h6>`
- `<header>`
- `<hgroup>`
- `<hr>`
- `<li>`
- `<main>`
- `<nav>`
- `<noscript>`
- `<ol>`
- `<output>`
- `<p>`
- `<pre>`
- `<section>`
- `<table>`
- `<tfoot>`
- `<ul>`
- `<video>`

### Elementos en línea

- `<a>`
- `<abbr>`
- `<acronym>`
- `<b>`
- `<bdo>`
- `<big>`
- `<button>`
- `<cite>`
- `<code>`
- `<dfn>`
- `<em>`
- `<g>`
- `<i>`
- `<img>`

- `<input>`
- `<kbd>`
- `<label>`
- `<map>`
- `<object>`
- `<samp>`
- `<script>`
- `<select>`
- `<small>`
- `<span>`
- `<strong>`
- `<sub>`
- `<sup>`
- `<textarea>`
- `<time>`
- `<tt>`
- `<var>`

## Etiquetas obsoletas

Con el paso del tiempo y la transición desde versiones anteriores a HTML5 (por ejemplo, desde HTML4 o XHTML), muchas etiquetas HTML han sido marcadas como [obsoletas](#).

Además de las etiquetas, también hay comportamientos que se consideran [obsoletos](#). Aunque su uso no es incorrecto, se desaconseja, como es el caso de indicar el atributo `type="text/css"` en un elemento `<style>`, ya que este tipo se infiere automáticamente en HTML5.

## Atributos comunes en HTML

Los atributos son palabras clave que modifican ligeramente el comportamiento de la etiqueta que los contienen.

Los [atributos comunes](#) son atributos que pueden utilizarse en prácticamente **cualquier etiqueta HTML**.

### Atributos CSS

- **id**: establece un identificador único para la etiqueta HTML. Debe ser un nombre único en todo el documento.
- **class**: establece una clase a una etiqueta HTML. Puede repetirse a lo largo del documento.
- **style**: aplica propiedades CSS directamente al elemento HTML en cuestión.

#### El atributo *id* (identificador)

En HTML, podemos asignar un identificador a una etiqueta y, de este modo, darle un nombre. Simplemente, añadimos el atributo `id` y colocamos el nombre como valor de ese atributo.

```
<div id="header">
  <div>Aquí irá un anuncio</div>
  <div id="articulo">Aquí irá el contenido de texto del artículo</div>
  <div>Aquí irá un anuncio</div>
</div>
```

Ese identificador único tiene unas normas:

- No debe empezar nunca por un número, pero puede contener números más adelante.

- El texto debe estar en *kebab-case*, es decir, en minúsculas y separado por guiones.
- Es preferible que no contenga caracteres especiales, acentuados o emojis.
- En un documento HTML no pueden existir dos elementos con el mismo id.

Los identificadores suelen utilizarse para zonas específicas que no se van a repetir, como **header** o **footer**.

La recomendación es utilizar clases en vez de identificadores **siempre que sea posible**.

### El atributo *class* (clase)

Las clases funcionan de una forma muy similar a los identificadores, pero son mucho más flexibles ya que no tiene la limitación de ser únicas. La idea de las clases es establecer géneros o tipos de etiquetas a las que les asociamos características comunes.

```
<div id="pagina">
  <div class="anuncio">Aquí irá un anuncio</div>
  <div id="articulo">Aquí irá el contenido de texto del artículo</div>
  <div class="anuncio ultimo">Aquí irá un anuncio</div>
</div>
```

La ventaja de utilizar clases es que, mediante CSS, se puede aplicar un estilo concreto a todas las etiquetas HTML que tengan esa clase.

Además, a diferencia de los identificadores, una etiqueta puede tener múltiples clases diferentes separadas por un espacio. Si se indican múltiples atributos `class` en la misma etiqueta, el navegador no ignorará los primeros; cada clase se aplicará y se combinará en el estilo final.

### El atributo *style* (estilos en línea)

El atributo `style` se utiliza en las etiquetas HTML para incrustar código CSS directamente en la propia etiqueta.

```
<p style="background: indigo; color: white;">Esto es un mensaje con estilos CSS</p>
```

En la mayoría de los casos, no se recomienda añadir estilos de esta forma, ya que suele considerarse una mala práctica. Es mejor colocar el código CSS en un archivo `.css` separado.

## Atributos de idioma

- **lang**: indica el idioma del contenido de la etiqueta HTML.
- **translate**: indica si el contenido de la etiqueta se debería traducir o no.
- **dir**: establece la direccionalidad del texto.

### El atributo *lang* (idioma)

Mediante el atributo `lang` podemos indicar el **idioma** del contenido de la etiqueta. El valor de este atributo debe ser el **código ISO 639-1** correspondiente al idioma que queremos especificar.

Aunque en principio podemos usar este atributo en cualquier etiqueta HTML, es **obligatorio** hacerlo en la etiqueta `<html>`, ya que esta etiqueta abarca todo el documento y establece el idioma del mismo:

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

```
...
</html>
```

Es completamente válido utilizar este atributo en una etiqueta específica para indicar un idioma diferente al especificado en la etiqueta `<html>` :

```
<!DOCTYPE html>
<html lang="es">
...
<p lang="it">È perfettamente valido utilizzare questo attributo su un'etichetta specifica</p>
...
</html>
```

### El atributo *translate* (traducción)

El atributo `translate` se puede utilizar en las etiquetas HTML y acepta los valores 'yes' y 'no'. Por defecto, si este atributo no se añade a una etiqueta, su valor es 'yes', lo que significa que todas las etiquetas son consideradas como «traducibles».

```
<p>
  Hace algunos días fuí a ver la nueva película de <span translate="no">StarWars</span>.
</p>
```

Con este atributo, podemos indicar a herramientas como *Google Translate* que no traduzcan una etiqueta que por ejemplo puede contener el título de un libro, el cual debe conservarse tal cual.

### El atributo *dir* (direccionalidad)

El atributo `dir` permite al desarrollador indicar la **direccionalidad del texto** en el documento. Esto es especialmente útil para idiomas que se escriben o leen de derecha a izquierda, en lugar de izquierda a derecha.

El **valor por defecto** de este atributo es `ltr` (*left to right*, de izquierda a derecha). Sin embargo, podemos modificarlo y establecer el valor `rtl` (*right to left*, de derecha a izquierda) o `auto` para permitir que el *user agent* detecte automáticamente la dirección de escritura.

```
<p lang="it" dir="auto">I contenuti dell'elemento sono esplicitamente isolati dal testo</p>
```

## Otros atributos comunes

- **title**: mensaje mostrado en un *tooltip* (aviso emergente) al mover el ratón encima.
- **data-\***: metadatos en la propia etiqueta. Se puede usar cualquier nombre con prefijo `data-`.
- **accesskey**: combinación de teclas que puede pulsar el usuario para activar el elemento.

### El atributo *title* (título)

Aunque se utiliza principalmente en las etiquetas de imágenes `<img>`, la mayoría de las etiquetas HTML pueden tener el atributo `title`, que especifica un mensaje de texto que aparece cuando el usuario detiene el ratón sobre el elemento un instante.

```
<figure>
  
```

```
<figcaption title="Pie de foto">One Web W3C for All</figcaption>
</figure>
```

Aunque los atributos `alt` y `title` en las etiquetas `<img>` tienen objetivos parecidos:

- El atributo `alt` debe ser un texto alternativo que describa la imagen (en caso de que no se pueda ver visualmente)
- El atributo `title` puede describir la imagen, pero no necesariamente debe ser una descripción alternativa.

### El atributo *data-* (metadatos)

En un documento HTML, la mayoría de los **metadatos** (información adicional) se incluyen dentro de la etiqueta `<head>`. Sin embargo, también se pueden incluir metadatos en etiquetas HTML a través de atributos que comienzan con el prefijo `data-`.

Los **atributos de datos personalizados** están destinados a almacenar datos personalizados, estados, anotaciones y similares, que son privados a la página o aplicación y para los cuales no hay atributos o elementos más apropiados.

Desde Javascript, si queremos hacer referencia a estos elementos, podemos hacerlo de varias formas:

```
<!-- HTML5 -->
<div class="spaceship" data-ship-id="92432"
    data-weapons="laser 2" data-shields="50%"
    data-x="30" data-y="10" data-z="90">
</div>
```

```
// JavaScript
const spaceship = document.getElementsByClassName("spaceship");

console.log(spaceship[0].dataset.weapons) // "Laser 2"
console.log(spaceship[0].getAttribute("data-shields")); // "50%"
```

En el ejemplo, accedemos a la propiedad `.dataset`, donde tendremos una lista de propiedades dependiendo de los diferentes atributos con prefijo `data-` que tenga el elemento. Por otro lado, también podemos utilizar el método `.getAttribute()` del DOM.

### El atributo *accesskey* (atajo)

En HTML, es posible añadir el atributo `accesskey` para indicar un atajo de teclado que puede pulsar el usuario para activar ese elemento. De esta forma, al pulsar la combinación `Alt + <tecla>` se activa ese elemento:

```
<form>
  <input accesskey="N" placeholder="Nombre (ALT+N)" /><!-- Campo de datos -->
  <input accesskey="A" placeholder="Apellidos (ALT+A)" /><!-- Campo de datos -->
</form>
```

⚠ Su uso está **desaconsejado** debido a que no está estandarizado entre sistemas operativos y navegadores.

## Estructura de un documento HTML

Un documento HTML debe estar **bien formado sintácticamente** para que el navegador pueda leerlo correctamente. A continuación se presenta un ejemplo básico de la estructura de un documento HTML:

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



```
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Título del documento</title>
</head>
<body>
  <!-- ... -->
</body>
</html>
```

## El `<!DOCTYPE>` (tipo de documento)

HTML es una aplicación **SGML** (Standard Generalized Markup Language), por lo que es necesario que la primera línea de un documento contenga la indicación del lenguaje de etiquetas utilizado.

El `<!DOCTYPE>` o **tipo de documento** es una declaración especial que se escribe en la primera línea del documento HTML y que indica el tipo de etiquetas que se están utilizando. Esta declaración no se considera una etiqueta HTML en sí.

Es una **declaración obligatoria**, aunque si no se incluye, la página web probablemente continuará visualizándose de manera correcta. Sin embargo, omitirla puede hacer que el navegador entre en lo que se llama **'Quirks Mode'** (modo peculiar o modo no estándar), que activa un modo de retrocompatibilidad con páginas antiguas, procesando muchas etiquetas HTML o propiedades CSS de manera diferente.

Para los **documentos HTML5**, la declaración se escribe de la siguiente manera:

```
<!DOCTYPE html>
```

En versiones anteriores, como **HTML4 o XHTML**, el tipo de documento se especificaba en la primera línea de una forma más compleja:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

## Etiqueta `<html>`

El **elemento** `<html>` es el **elemento raíz** de un documento HTML. Se coloca inmediatamente después de la declaración del tipo de documento y abarca todo el contenido del documento.

Aunque el uso del atributo `lang` no es obligatorio, es altamente recomendable, ya que permite indicar al navegador el idioma del contenido. Este atributo es útil para los motores de búsqueda y para navegadores con síntesis de voz, facilitando el acceso a personas con discapacidad visual.

## Etiqueta `<head>`

La **etiqueta** `<head>` de un documento HTML se conoce comúnmente como **cabecera HTML**. Sin embargo, es importante destacar que esta cabecera no es una parte visual de la página web; más bien, es una sección del código HTML donde se incluyen etiquetas de **metadatos**. Estos metadatos establecen información sobre el documento que no necesariamente se visualiza en la página.

Algunos ejemplos de metadatos que se pueden incluir en la etiqueta `<head>` son:

- Título de la página (definido con `<title>`).

- Descripción de la página (generalmente a través de la etiqueta `<meta name="description">` ).
- Icono o *favicon* (definido con `<link rel="icon">` ).
- Configuración de la codificación de caracteres (definida con `<meta charset="UTF-8">` ).
- Configuraciones de *viewport* para dispositivos móviles (definida con `<meta name="viewport">` ).

```
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Título de la Página</title>
  <link rel="icon" href="favicon.ico" type="image/x-icon">
  <meta name="description" content="Descripción breve de la página.">
</head>
```

## Etiqueta `<title>`

La etiqueta `<title>` es **obligatoria** en un documento HTML y **sólo puede haber una** de este tipo por documento.

Esta etiqueta define el **título del documento**, que aparece en la barra de título de la ventana del navegador o en las pestañas. Además, el contenido de la etiqueta `<title>` se utiliza como enlace en los resultados de los motores de búsqueda, lo que la convierte en un elemento crucial para la optimización en motores de búsqueda (SEO).

## Etiqueta `<meta>`

La etiqueta `<meta>` se utiliza para almacenar metadatos del documento. Se pueden incluir varios elementos `<meta>` en la sección `<head>` , y no es obligatorio que haya al menos uno.

Uno de los metadatos más importantes es el que indica el tipo de codificación del documento. Es crucial definir la codificación justo después de la etiqueta de apertura `<head>` , ya que esta afectará a todos los elementos subsiguientes.

En HTML5, la **codificación por defecto es UTF-8**.

**Unicode es un estándar** que asigna un número único a cada carácter, lo que permite representar texto de manera uniforme. Por ejemplo, la letra "A" se representa con el número 65. Por su parte, **UTF-8 es un estándar de codificación** que especifica cómo estos números se traducen a un formato binario que puede ser almacenado y procesado por computadoras.

```
<head>
  <meta charset="UTF-8">
  <meta name="description" content="Descripción del documento">
  <meta name="author" content="Autor del documento">
</head>
```

## Etiqueta `<link>`

La etiqueta `<link>` se utiliza para crear enlaces a recursos externos que el navegador procesará, como hojas de estilo y *favicons*. Esta etiqueta se coloca dentro de la sección `<head>` del documento HTML:

```
<head>
  <link rel="stylesheet" href="/home.css">
  <link rel="icon" href="favicon.ico">
</head>
```

La etiqueta `<link>` también permite especificar diferentes tamaños y tipos de *favicon* mediante el atributo `rel` :

```
<head>
<title>Forums – Inbox</title>
<link rel=icon href=favicon.png sizes="16x16" type="image/png">
<link rel=icon href=windows.ico sizes="32x32 48x48" type="image/vnd.microsoft.icon">
<link rel=icon href=mac.icns sizes="128x128 512x512 8192x8192 32768x32768">
<link rel=icon href=iphone.png sizes="57x57" type="image/png">
<link rel=icon href=gnome.svg sizes="any" type="image/svg+xml">
<!-- ... -->
</head>
```

## Etiqueta `<style>`

La etiqueta `<style>` se utiliza para declarar estilos CSS que se aplican exclusivamente al documento actual. Se debe incluir dentro de la sección `<head>` del documento HTML.

```
<head>
<style>
  .autor {
    text-transform: uppercase; /* Convierte el texto a mayúsculas */
  }
</style>
</head>
```

No es necesario especificar el atributo `type="text/css"` en la etiqueta `<style>` , ya que CSS es considerado el tipo de contenido por defecto.

Aunque se pueden definir estilos directamente en el documento HTML, se recomienda utilizar hojas de estilo externas para mantener una mejor organización y reutilización del código.

## Etiqueta `<script>`

La etiqueta `<script>` se utiliza para incluir scripts de JavaScript que se aplican exclusivamente al documento actual. Esta etiqueta puede colocarse en cualquier parte del documento HTML, aunque se recomienda incluirla al final del cuerpo ( `<body>` ) para asegurar que el navegador procese el contenido HTML antes de ejecutar el script.

```
<script>
  alert ("Hello World!");
</script>
```

No es necesario especificar el atributo `type="text/javascript"` en la etiqueta `<script>` , ya que JavaScript es considerado el tipo de contenido por defecto.

Para utilizar un script desde un archivo externo, puede usarse el atributo `src` :

```
<script src="script.js"></script>
```

## Etiqueta `<body>`

La etiqueta `<body>` es esencial en un documento HTML, ya que incluye todos los elementos que constituyen el contenido visible de la página.

La apertura de la etiqueta `<body>` se sitúa justo después del cierre de la etiqueta del encabezado `</head>` :

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Título del documento</title>
</head>
<body>
  <h1>Bienvenido a mi página web</h1>
  <p>Este es un párrafo de ejemplo.</p>
  
  <a href="https://www.ejemplo.com">Visita nuestro sitio</a>
</body>
</html>
```

## Contenedores semánticos

Los contenedores en HTML permiten **organizar y estructurar el contenido de una página** de forma lógica. Estos contenedores pueden incluir una amplia gama de elementos como texto, imágenes, formularios, etcétera...

- **Contenedores genéricos**: no tienen un significado específico en cuanto al contenido que agrupan.
  - `<div>` : se utiliza para agrupar bloques de contenido. No tiene significado semántico y se usa principalmente para aplicar estilos o scripts.
  - `<span>` : similar a `<div>` , pero se utiliza para agrupar contenido en línea, como partes de texto, dentro de un párrafo.
- **Contenedores semánticos**: proporcionan información adicional sobre el tipo de contenido que contienen, lo que mejora la accesibilidad y la comprensión tanto para los navegadores como para los motores de búsqueda.
  - `<header>` : representa un encabezado de una página o sección.
  - `<footer>` : representa el pie de una página o sección.
  - `<article>` : define un contenido independiente y autónomo, como un artículo de una revista o una entrada de blog.
  - `<section>` : agrupa contenido temáticamente relacionado.
  - `<nav>` : define un conjunto de enlaces de navegación.
  - `<aside>` : contiene información relacionada pero tangencial al contenido principal, como una barra lateral.

### Elemento `<div>`

El elemento `<div>` es un contenedor genérico y uno de los elementos más antiguos en HTML. Se utiliza para agrupar otros elementos y contenido, permitiendo incluir cualquier tipo de elementos, incluidos otros `<div>` . Sin embargo, su uso no proporciona información semántica sobre el contenido.

Aunque `<div>` es útil para estructurar un documento, es preferible utilizar contenedores semánticos como `<header>` , `<footer>` , o `<section>` , que describen mejor la naturaleza del contenido.

```
<!-- Contenedores neutros -->
<div id="article">
  <div id="header">
    <!-- Título -->
```

```

</div>
<div id="main">
  <!-- Cuerpo -->
</div>
<div id="footer">
  <!-- Pie -->
</div>
</div>

<!-- Contenedores semánticos -->
<article>
  <header>
    <!-- Título -->
  </header>
  <p><!-- Cuerpo --></p>
  <footer>
    <!-- Pie -->
  </footer>
</article>

```

💡 El elemento `<div>` es un elemento de bloque.

## Elemento `<span>`

El elemento `<span>` es un contenedor genérico que se utiliza normalmente para formatear de manera particular un texto dentro de un párrafo de texto, como por ejemplo:

```

<style>
  .fondo-gris{
    background-color: #eee;
  }
</style>

<p>Unde consequatur amet itaque. Velit <span class="fondo-gris">rerum sed iusto</span> quae
consectetur voluptas temporibus.</p>

```

💡 El elemento `<span>` es un elemento en línea.

## Elemento `<header>`

El elemento `<header>` es un contenedor semántico que se utiliza para definir una sección introductoria o de encabezado, y suele contener elementos como títulos, logotipos o enlaces de navegación. Se puede usar en diferentes niveles:

- A nivel de **página**: para mostrar encabezados generales como un logotipo o una barra de navegación.
- A nivel de **contenido**: dentro de un artículo u otra sección para definir un encabezado específico.

```

<article>
  <header>
    <h2>Encabezado del artículo</h2>
  </header>
  <p>Cuerpo del artículo.</p>
  <footer>
    <p>Pie del artículo.</p>
  </footer>
</article>

```

Desde HTML5.1, es posible anidar elementos `<header>` y `<footer>` dentro de otro elemento `<header>` si los dos primeros elementos se incluyen en el mismo elemento padre:

```

<article>
  <header>
    <h2>Título del artículo</h2>
    <aside>
      <header>
        <h3>Título del contenido secundario</h3>
      </header>
      <p>Texto adicional dentro del artículo.</p>
      <footer>
        <p>Pie del contenido secundario.</p>
      </footer>
    </aside>
  </header>
  <p>Texto principal del artículo.</p>
</article>

```

💡 El elemento `<header>` es un elemento de bloque.

## Elemento `<footer>`

El elemento `<footer>` es un contenedor genérico que se utiliza para definir una sección de pie de página en un documento o en una parte específica del contenido, como un artículo o una sección. Al igual que el `<header>`, puede emplearse a diferentes niveles:

- A nivel de **página**: contiene información como derechos de autor, enlaces de contacto o políticas de privacidad.
- A nivel de **contenido**: por ejemplo, al final de un artículo, se puede utilizar para mostrar información adicional relacionada con ese contenido.

```

<article>
  <header>
    <h2>Título del artículo</h2>
  </header>
  <p>Contenido del artículo.</p>
  <footer>
    <p>© 2024 Autor del artículo</p>
  </footer>
</article>

```

El elemento `<footer>` se puede utilizar independientemente del uso de un `<header>`, y es común verlo al final de la página o en la conclusión de una sección de contenido.

💡 El elemento `<footer>` es un elemento de bloque.

## Elemento `<aside>`

El elemento `<aside>` es un contenedor semántico que se utiliza para definir contenido relacionado o complementario al contenido principal. Es común emplearlo en barras laterales, anuncios, citas destacadas o cualquier información que apoye el contenido principal sin ser parte directa de él.

```

<article>
  <h2>Artículo principal</h2>
  <p>Este es el contenido principal del artículo.</p>
  <aside>
    <h3>Nota relacionada</h3>
    <p>Esta es una información adicional o relacionada con el artículo principal.</p>
  </aside>

```

```
</aside>
</article>
```

💡 El elemento `<aside>` es un elemento de bloque.

## Elemento `<nav>`

El elemento `<nav>` es un contenedor semántico que define una sección del documento destinada a la navegación. Suele contener enlaces a diferentes secciones del mismo documento o a otros sitios web.

```
<nav>
  <ul>
    <li><a href="#inicio">Inicio</a></li>
    <li><a href="#servicios">Servicios</a></li>
    <li><a href="#contacto">Contacto</a></li>
  </ul>
</nav>
```

Es importante destacar que no todo conjunto de enlaces tiene que estar dentro de un elemento `<nav>`. Sólo aquellos que son parte de la navegación principal o estructural del sitio deberían usarlo.

💡 El elemento `<nav>` es un elemento de bloque.

## Elemento `<main>`

El elemento `<main>` es un contenedor semántico que se utiliza para contener el **contenido principal** de un documento HTML, es decir, el contenido que es único y específico de la página. Este contenido debe ser el más relevante y significativo para la estructura del sitio.

```
<main>
  <h1>Título Principal</h1>
  <p>Este es el contenido más importante de la página.</p>
</main>
```

Este elemento debe ser único en el documento. Sólo puede haber un elemento `<main>` en el documento.

Además, no debe estar dentro de otros elementos de estructura como `<header>`, `<footer>`, `<nav>`, `<aside>` o `<article>`.

💡 El elemento `<main>` es un elemento de bloque.

## Elemento `<section>`

El elemento `<section>` es un contenedor semántico que se utiliza para **agrupar contenido relacionado** que comparte un tema o propósito común. Cada `<section>` debe contener un encabezado (generalmente un `<h1>`, `<h2>`, etc.) que identifique el tema de esa sección, así como la posibilidad de incluir un pie de página.

```
<section>
  <h2>Título de la Sección</h2>
  <p>Este es el contenido de la sección que trata sobre un tema específico.</p>
  <footer>
    <p>Información adicional o referencias.</p>
  </footer>
</section>
```

Este elemento facilita la organización del documento en partes claramente definidas y ayuda a los lectores de pantalla a entender mejor la estructura del contenido.

Es recomendable que cada `<section>` contenga su propio encabezado, para que quede claro cuál es el tema que abarca.

La utilización de varios elementos `<section>` facilita estructurar un documento en secciones distintas.

💡 El elemento `<section>` es un elemento de bloque.

## Elemento `<article>`

El elemento `<article>` es un contenedor semántico diseñado para representar un **contenido autónomo y reutilizable**. Como indica su nombre, su uso más habitual es la creación de artículos en un blog.

```
<article>
  <header>
    <h2>Título del Artículo</h2>
    <p>Fecha de publicación: <time datetime="2024-09-24">24 de septiembre de 2024</time></p>
  </header>
  <p>Este es el contenido, que es completamente autónomo y tiene su propio sentido.</p>
  <footer>
    <p>Escrito por: Autor del artículo</p>
  </footer>
</article>
```

💡 El elemento `<article>` es un elemento de bloque.

## Elemento `<details>`

El elemento `<details>` es un contenedor semántico que muestra **información adicional** de forma interactiva. Este elemento es útil para revelar o esconder contenido cuando el usuario lo solicita, mejorando así la experiencia del usuario al interactuar con la página.

El elemento `<details>` contiene un elemento `<summary>`, que actúa como el título o encabezado del contenido que puede expandirse. Al hacer clic en el `<summary>`, se despliega el contenido adicional, que permanece oculto hasta que el usuario decide visualizarlo.

```
<section class="progress window">
  <h1>Copying "Really Achieving Your Childhood Dreams"</h1>
  <details>
    <summary>Copying... <progress max="100" value="25"></progress> 25%</summary>
    <dl>
      <dt>Transfer rate:</dt> <dd>452KB/s</dd>
      <dt>Local filename:</dt> <dd>/home/rpausch/raycd.m4v</dd>
      <dt>Remote filename:</dt> <dd>/var/www/lectures/raycd.m4v</dd>
      <dt>Duration:</dt> <dd>01:16:27</dd>
      <dt>Color profile:</dt> <dd>SD (6-1-6)</dd>
      <dt>Dimensions:</dt> <dd>320x240</dd>
    </dl>
  </details>
</section>
```

💡 El elemento `<details>` es un elemento de bloque.

## Contenedores de texto



Los contenedores de texto son elementos semánticos en HTML diseñados específicamente para mostrar y organizar el **contenido textual**. Cada uno de estos contenedores tiene un propósito definido y se utiliza para estructurar el texto en el documento, mejorando la legibilidad del contenido.

Todos los contenedores son de tipo **bloque**, lo que implica que utilizarán toda la longitud del contenedor padre. Cada uno de estos elementos comienza en una nueva línea automáticamente.

Además, los navegadores insertan un espacio antes y después de estos contenedores. Este comportamiento se puede modificar mediante reglas CSS.

## Elementos `<hx>`

Los [elementos de título](#) en HTML permiten insertar hasta **seis niveles de títulos jerárquicos**, organizados desde el más importante ( `<h1>` ) hasta el menos importante ( `<h6>` ).

Los elementos de título son esenciales para mejorar la accesibilidad y el SEO, ya que permiten a los motores de búsqueda y lectores de pantalla interpretar mejor la estructura del contenido.

Es aconsejable **no saltarse ningún nivel** para mantener la coherencia del documento.

Además, es perfectamente válido repetir niveles de jerarquía en contenedores distintos. Por ejemplo, es posible utilizar un título `<h2>` en varios contenedores `<section>` .

```
<body>
  <h1>Let's call it a draw(ing surface)</h1>
  <section>
    <h2>Diving in</h2>
  </section>
  <section>
    <h2>Simple shapes</h2>
  </section>
  <section>
    <h2>Canvas coordinates</h2>
    <section>
      <h3>Canvas coordinates diagram</h3>
    </section>
  </section>
  <section>
    <h2>Paths</h2>
  </section>
</body>
```

💡 Los elementos `<hx>` son elementos de bloque.

## Elemento `<p>`

El [elemento `<p>`](#) se utiliza para definir un **párrafo de texto** en HTML. Este elemento organiza el contenido textual en bloques separados, lo que mejora la legibilidad y la estructura del documento.

Cada párrafo se considera un bloque de texto autónomo y siempre comenzará en una nueva línea.

```
<p>The little kitten gently seated herself on a piece of carpet.</p>

<fieldset>
  <legend>Personal information</legend>
  <p>
    <label>Name: <input name="n"></label>
    <label><input name="anon" type="checkbox"> Hide from other users</label>
  </p>
```

```
<p><label>Address: <textarea name="a"></textarea></label></p>
</fieldset>
```

💡 El elemento `<p>` es un elemento de bloque.

## Elemento `<blockquote>`

El elemento `<blockquote>` se utiliza para mostrar citas o textos largos extraídos de fuentes externas. Este elemento añade una sangría o un estilo especial para destacar el contenido citado del texto normal.

Es un contenedor flexible. Puede incluir otros elementos como títulos, párrafo, imagen, etc...

Se puede especificar el origen de la cita mediante el atributo `cite`.

```
<blockquote>
  <p>[Jane] then said she liked [...] fish.</p>
</blockquote>
```

💡 El elemento `<blockquote>` es un elemento de bloque.

## Elemento `<address>`

El elemento `<address>` se utiliza para representar la información de contacto de una persona, empresa u organización dentro de un documento. Se puede utilizar en diversas partes del documento, generalmente dentro de un pie de página.

Este elemento puede contener otros elementos en su interior como enlaces, etc...

```
<footer>
  <address>
    For more details, contact
    <a href="mailto:js@example.com">John Smith</a>.
  </address>
  <p><small>© copyright 2038 Example Corp.</small></p>
</footer>
```

💡 El elemento `<address>` es un elemento de bloque.

## Elemento `<pre>`

El elemento `<pre>` se utiliza para mostrar texto preformateado, es decir, conserva los espacios en blanco, saltos de línea y tabulaciones tal como se escriben en el código fuente. A diferencia de otros elementos HTML, no ignora los espacios en blanco. Es ideal para representar fragmentos de código o cualquier otro tipo de texto que requiera una estructura específica.

Se usa frecuentemente junto con el elemento `<code>` para mostrar fragmentos de código.

```
<p>This is the <code>Panel</code> constructor:</p>
<pre>
  <code>
    function Panel(element, canClose, closeHandler) {
      this.element = element;
      this.canClose = canClose;
      this.closeHandler = function () { if (closeHandler) closeHandler() };
    }
  </code>
</pre>
```

```
</code>
</pre>
```

💡 El elemento `<pre>` es un elemento de bloque.

## Elemento `<hr>`

El elemento `<hr>` inserta una línea horizontal en el documento y se utiliza como un **divisor visual** entre secciones de contenido, facilitando la separación de ideas o bloques de texto dentro de una página web.

Se puede personalizar con CSS para ajustar su estilo, grosor, color, etc.

💡 El elemento `<hr>` es un elemento de bloque.

## Elemento `<ul>`

El elemento `<ul>` se utiliza para crear **listas no ordenadas** (*unordered lists*) en HTML. Los elementos de la lista se presentan con una viñeta o símbolo delante de cada ítem.

Cada ítem de la lista se define con el elemento `<li>`, lo que lo convierte en un bloque individual dentro de la lista.

Se utiliza comúnmente para barras de navegación y listas de elementos de cualquier tipo, donde el orden no es importante.

⚠ La etiqueta de cierre `</li>` **puede omitirse** si a continuación hay otro elemento `<li>` o no hay más contenido en el elemento padre.

```
<p>I have lived in the following countries:</p>
<ul>
  <li>Norway
  <li>Switzerland
  <li>United Kingdom
  <li>United States
</ul>
```

💡 El elemento `<ul>` es un elemento de bloque.

## Elemento `<menu>`

El elemento `<menu>` es una alternativa semántica al uso de listas no ordenadas ( `<ul>` ) para representar **menús o barras de herramientas**, donde cada elemento representa un comando o acción que el usuario puede ejecutar o activar.

Es útil en interfaces que contienen opciones de comandos o botones, como menús contextuales o barras de herramientas.

Cada opción del menú se define con un elemento `<li>`, que puede contener botones, enlaces u otros elementos interactivos.

```
<menu>
  <li><button onclick="copy()"></button></li>
  <li><button onclick="cut()"></button></li>
  <li><button onclick="paste()"></button></li>
</menu>
```

💡 El elemento `<menu>` es un elemento de bloque.

## Elemento `<ol>`

El elemento `<ol>` se utiliza para crear **listar ordenadas** (*ordered list*) en HTML. Los elementos de la lista se presentan con una **cifra** delante de cada ítem.

Cada ítem de la lista se define utilizando el elemento `<li>`, y este puede incluir el atributo `value` para especificar un valor personalizado para el número que aparecerá al inicio del ítem.

⚠ La etiqueta de cierre `</li>` **puede omitirse** si a continuación hay otro elemento `<li>` o no hay más contenido en el elemento padre.

```
<p>I have lived in the following countries:</p>
<ol>
  <li>Switzerland
  <li>United Kingdom
  <li value="5">United States
  <li value="8">Norway
</ol>
```

El elemento `<ol>` tiene varios atributos:

- `start` : define el valor inicial de la numeración.
- `reversed` : valor booleano que invierte el orden de la lista, comenzando por el último ítem.
- `type` : cambia el tipo de enumeración con los siguientes valores:
  - `type="1"` : números decimales
  - `type="a"` : letras minúsculas
  - `type="A"` : letras mayúsculas
  - `type="i"` : números romanos en minúsculas
  - `type="I"` : números romanos en mayúsculas

```
<p>I have lived in the following countries</p>
<ol reversed start="5" type="a">
  <li>Switzerland
  <li>United Kingdom
  <li>United States
  <li>Norway
</ol>
```

💡 El elemento `<ol>` es un elemento de bloque.

## Elemento `<dl>`

El elemento `<dl>` permite crear listas de definiciones, ideales para describir términos específicos.

Para crear una lista de definiciones hay tres elementos disponibles:

- `<dl>` : define la lista de definiciones (*description list*).
- `<dt>` : indica el término que se va a definir (*description term*).
- `<dd>` : proporciona la definición del término (*description definition*), que se identifica respecto al término.

```
<dl>
  <dt lang="en-US"> <dfn>color</dfn> </dt>
  <dt lang="en-GB"> <dfn>colour</dfn> </dt>
  <dd> A sensation which (in humans) derives from the ability of
  the fine structure of the eye to distinguish three differently
```

```
filtered analyses of a view. </dd>
</dl>
```

💡 El elemento `<dl>` es un elemento de bloque.

## Formateo del texto

El formateo de texto en HTML se puede lograr mediante una variedad de etiquetas, que **se dividen en semánticas y no semánticas**. Las etiquetas semánticas proporcionan información sobre el contenido, mientras que las no semánticas, que provienen de versiones anteriores de HTML, no lo hacen.

### Formateo semántico del texto

El formateo semántico permite resaltar palabras en un contenedor de tipo bloque con elementos en línea.

- `<strong>` : marcado semántico que indica especial énfasis, presentando el texto en negrita (no solo como resaltado).
- `<em>` : marcado semántico que aplica énfasis sencillo, mostrando el texto en cursiva.
- `<ins>` : marcado semántico para indicar texto que debe ser subrayado.
- `<del>` : marcado semántico que muestra texto como tachado.
- `<sub>` y `<sup>` : permiten colocar caracteres como índices o exponentes, respectivamente.
- `<small>` : marcado semántico para mostrar un texto más pequeño.
- `<cite>` y `<q>` : marcado para indicar el título de una obra (en cursiva) y una cita de dicha obra (entre comillas dobles).
- `<dfn>` : marcado semántico para definir un término.
- `<abbr>` : marcado semántico para representar una abreviatura.
- `<code>` : marcado semántico para indicar código informático.
- `<mark>` : marcado semántico que destaca un texto específico.
- `<br>` : permite cambiar de línea dentro de un párrafo permaneciendo estructuralmente en el mismo párrafo

### Formateo no semántico

Estas etiquetas permiten el formateo del texto pero no brindan información semántica. Se recomienda utilizar siempre sus alternativas semánticas para proporcionar más contexto al navegador y al usuario.

Por ejemplo, es recomendable utilizar la etiqueta `<strong>` antes que la etiqueta `<b>` para resaltar un texto.

- `<b>` : resaltado en negrita sin información semántica
- `<i>` : resaltado en cursiva sin información semántica
- `<u>` : resaltado con subrayado sin información semántica
- `<s>` : resaltado con tachado sin información semántica

## Caracteres especiales

En HTML, los caracteres especiales como flechas o símbolos se representan con **una notación de entidad**, que tiene la forma `&{code};` . Esto es especialmente útil para incluir caracteres que de otro modo podrían ser interpretados como parte del código HTML.

Un ejemplo común es el espacio de no separación (*Non-breaking space*), que se denota como `&nbsp;` . Este carácter permite crear un espacio que no se puede dividir con un salto de línea.

Algunos recursos útiles para caracteres especiales en HTML incluyen:

- [HTML Entities](#): una lista completa de entidades HTML para utilizar.
- [HTML Symbols](#): un recurso que proporciona una variedad de símbolos disponibles en HTML.
- [Using Emojis in HTML](#): instrucciones sobre cómo utilizar emojis en documentos HTML.

Ejemplos de caracteres especiales:

- `&nbsp;`
- `&rarr;` = →
- `&larr;` = ←
- `&copy;` = ©
- `&hearts;` = ♥
- `&lt;` = <
- `&gt;` = >

## Los enlaces

---

La Web esta formada por [enlaces](#), que representan una conexión entre dos recursos, siendo uno de ellos el propio documento.

En un documento HTML hay tres tipos de enlaces:

- **Enlaces a recursos externos** como hojas de estilo CSS o ficheros JavaScript que serán procesados por el *user agent* o navegador.
- **Enlaces de hipertexto** que son enlaces a otros recursos y que son accesibles mediante la interacción del usuario (como pulsar sobre ellos).
- **Enlaces a recursos internos** son enlaces a recursos dentro del documento actual, utilizados para proporcionar significado o comportamiento especial.

## Elemento `<a>`

Los enlaces se crean con la [etiqueta](#) `<a>` aunque existen otras etiquetas como `<area>` , `<form>` y `<link>` que son enlaces especiales.

Atributos aplicables a los enlaces:

- `href` : obligatorio para indicar la URL de destino.
- `hreflang` : especifica el idioma de destino.
- `rel` : indica el tipo de relación del enlace que se establece.
- `target` : especifica el contexto de apertura del enlace:
  - `target="_blank"` : el destino se abre en una nueva ventana o una nueva pestaña
  - `target="_parent"` : el destino se abre en el elemento padre.
  - `target="_self"` : el destino se abre en el mismo navegador. No hay cambio de contexto.
  - `target="_top"` : el destino se abre en el padre superior en la jerarquía de ventanas.

Los enlaces son **relativos** cuando apuntan a un documento del mismo sitio web, mientras que un enlace es **absoluto** si apunta a otro sitio web.

```
<nav>
<ul>
<li><a href="/">Home</a></li> <!--Enlace relativo -->
<li><a href="/news">News</a></li> <!--Enlace relativo -->
<li><a href="/legal">Legal</a></li> <!--Enlace relativo -->
<li><a href="#sectionA">Section A</a></li> <!--Enlace interno -->
<li><a href="#sectionB">Section B</a></li> <!--Enlace interno -->
<li><a href="http://google.es">Buscador</a></li> <!--Enlace absoluto -->
</ul>
</nav>
<section id="sectionA">
<h2>Section A</h2>
<p>Lorem ipsum dolor sit amet consectetur adipisicing elit.</p>
</section>
<section id="sectionB">
<h2>Section B</h2>
<p>Lorem ipsum dolor sit amet consectetur adipisicing elit.</p>
</section>
```

Los enlaces **internos** al propio documento facilitan la navegación al usuario cuando el documento tiene una longitud elevada. Para implementar un enlace interno es necesario que el destino del enlace tenga el atributo `id`.

💡 El elemento `<a>` es un elemento en línea.

## Tablas

Las tablas se utilizan para mostrar datos tabulares. Cualquier otro uso, como estructurar un documento, no es correcto. El elemento padre para contener una tabla es el elemento `<table>`.

Elementos de una tabla:

- `<tr>` : permite insertar filas en la tabla (*table row*).
- `<td>` : permite crear celdas en cada fila (*table data*).
- `<th>` : permite crear celdas con encabezado (*table header*).
- `<caption>` : permite añadir un título a la tabla.

```
<table>
<caption>Resultados primer trimestre</caption>
<tr>
<th>&nbsp;</th>
<th>Enero</th>
<th>Febrero</th>
<th>Marzo</th>
</tr>
<tr>
<th>Nantes</th>
<td>1.84M</td>
<td>2.33M</td>
<td>1.39M</td>
</tr>
<tr>
<th>París</th>
<td>4.24M</td>
<td>2.29M</td>
<td>5.17M</td>
```

```
</tr>
</table>
```

Las etiquetas de cierre `</tr>`, `</td>`, `</th>` y `</caption>` se pueden omitir en determinados supuestos.

Para fusionar celdas horizontalmente se utiliza el atributo `colspan` mientras que para fusionar celdas verticalmente se utiliza el atributo `rowspan`. En ambos casos se indica el número de celdas a fusionar.

Para estructurar la tabla se pueden usar las etiquetas `<thead>`, `<tbody>` y `<tfoot>` de forma que se puedan agrupar las filas según su significado.

```
<table>
  <caption>Resultados primer trimestre</caption>
  <thead>
    <tr>
      <th>&nbsp;</th>
      <th>Enero</th>
      <th>Febrero</th>
      <th>Marzo</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <th>Nantes</th>
      <td>1.84M</td>
      <td>2.33M</td>
      <td>1.39M</td>
    </tr>
    <tr>
      <th>París</th>
      <td>4.24M</td>
      <td>2.29M</td>
      <td>5.17M</td>
    </tr>
  </tbody>
  <tfoot>
    <tr>
      <th>Total</th>
      <td>6.08M</td>
      <td>4.62M</td>
      <td>6.56M</td>
    </tr>
  </tfoot>
</table>
```

## Imágenes

Las **imágenes** son elementos fundamentales en la web, ya que enriquecen el contenido visual y mejoran la experiencia del usuario. En HTML, las imágenes se insertan normalmente utilizando la etiqueta `<img>`, que permite incluir diversos formatos de imagen como JPEG, PNG y GIF.

### Elemento `<img>`

Para insertar una imagen en un documento HTML se utiliza la etiqueta `<img>`.

El atributo `src` es **obligatorio** para indicar la ruta de acceso a la imagen a mostrar. Esta ruta puede ser relativa al propio sitio web o absoluta.

Con el atributo `alt` se puede mostrar un texto alternativo si la imagen no se puede cargar.



Los atributos `width` y `height` permiten indicar el espacio asignado a la visualización de la imagen. Si no se informan, el navegador debe esperar a la carga del archivo de imagen para determinar sus dimensiones y reservar el espacio. En caso de que el tamaño de alto o ancho de la imagen difieran de los indicados en los atributos, éstos tendrán preferencia.

## Elemento `<picture>`

El elemento `<picture>` permite mostrar diferentes imágenes para diferentes dispositivos o tamaños de pantalla. Este **elemento actúa como contenedor** y ofrece una manera flexible de manejar imágenes responsivas.

Dentro del contenedor `<picture>` podemos utilizar etiquetas `<img>` o `<source>` para especificar diferentes imágenes a través del atributo `srcset`. El navegador elegirá automáticamente el primer elemento que mejor se ajuste a las características del dispositivo del usuario.

Es recomendable especificar una etiqueta `<img>` al final del contenedor `<picture>` que será utilizada por los navegadores que no soporten la etiqueta `<picture>`.

```
<picture>
  <source media="(min-width: 650px)" srcset="img_food.jpg">
  <source media="(min-width: 465px)" srcset="img_car.jpg">
  
</picture>
```

## Formularios

Los **formularios** son una parte esencial de la interacción en la web, permitiendo a los usuarios enviar información y datos a los servidores para su procesamiento.

Los formularios constan de diferentes tipos de controles, como campos de texto, botones, casillas de verificación, menús desplegables, selectores, etc..., brindando una forma versátil de recopilar información.

## Atributos más utilizados

Los elementos HTML relacionados con formularios pueden utilizar diversos atributos que controlan su comportamiento y apariencia. A continuación se detallan algunos de los atributos más comunes y útiles para gestionar la funcionalidad de los campos de entrada, aunque no todos ellos se aplican a todos los tipos de elementos de formulario:

- `id` : identifica de forma única el campo dentro del formulario.
- `name` : define el nombre del campo, necesario para enviar el valor al hacer *submit* del formulario y que se utilizará para recuperar los datos enviados.
- `autocomplete` : permite el autocompletado de los campos.
- `autofocus` (booleano): activa inmediatamente el campo al cargar la página.
- `disabled` (booleano): desactiva el campo, impidiendo su uso por parte del usuario.
- `placeholder` : muestra un texto indicativo dentro del campo.
- `readonly` (booleano): marca el campo como de solo lectura.
- `required` (booleano): indica que el campo es obligatorio para el envío del formulario.

## Elemento `<form>`

El elemento `<form>` es el contenedor de un formulario. Este elemento tiene varios atributos:

- `action` : la URL del script que va a hacerse cargo de los datos introducidos en el formulario.
- `method` : especifica si los datos se enviarán usando HTTP con el método `'POST'` o `'GET'` .
- `name` : asigna un nombre al formulario.
- `enctype` : indica el tipo MIME de los datos enviados:
  - `"application/x-www-form-urlencoded"` : formato por defecto. Los datos se codifican como pares clave-valor.
  - `"multipart/form-data"` : para el envío de archivos.

```
<form action="backend.php" method="post" name="inscription"
      enctype="application/x-www-form-urlencoded">
  ...
</form>
```

## Elemento `<fieldset>`

El elemento `<fieldset>` se utiliza para **agrupar visualmente los campos** de un formulario, mejorando así la organización y legibilidad del contenido. Es especialmente útil cuando tienes varios grupos de campos relacionados.

Se puede añadir una leyenda o título que describa el propósito del grupo de campos con el elemento `<legend>` .

```
<form method="post" action="order.cgi">
  <fieldset>
    <legend>Personal data</legend>
    <p><label>Full name: <input name="fn"> <small>Format: First Last</small></label></p>
    <p><label for="age">Age: <input min="0" name="age" type="number"></label></p>
    <p><label>Post code: <input name="pc"> <small>Format: AB12 3CD</small></label></p>
  </fieldset>
</form>
```

## Elemento `<label>`

El elemento `<label>` se utiliza para **asociar una etiqueta descriptiva a un campo** de formulario. Cuando el usuario pulsa en la etiqueta, el campo asociado se activa automáticamente, facilitando la interacción con el formulario.

Este elemento facilita la utilización de los formularios y facilita la accesibilidad a las personas con discapacidad visual, ya que los lectores de pantalla leen la etiqueta asociada antes de los campos del formulario.

Hay dos formas de asociar el elemento `<label>` con el campo del formulario:

1. Asocia la etiqueta `<label>` con el campo mediante el valor del atributo `for` , que debe coincidir con el atributo `id` del campo correspondiente.
2. Encapsular el campo del formulario dentro del `<label>` , por lo que ya no es necesario usar el atributo `for` ni el `id` .

Ambas formas son válidas y aceptadas en HTML.

```
<form method="post" action="order.cgi">
  <!-- Usando el atributo for e id -->
  <p><label for="fullName">Nombre completo: <input id="fullName" name="fn"></label></p>
```

```
<!-- Encapsulando el campo dentro del label -->
<p><label>Edad: <input name="age" type="number" min="0"></label></p>
<p><label>Código postal: <input name="pc"> <small>Formato: AB12 3CD</small></label></p>
</form>
```

Elementos que pueden 'etiquetables' con la etiqueta `<label>` :

- `<button>`
- `<input>` (si no están en el estado 'hidden')
- `<meter>`
- `<output>`
- `<progress>`
- `<select>`
- `<textarea>`
- `<button>`
- elementos personalizados

## Elemento `<input>`

El elemento `<input>` es uno de los elementos más versátiles de los formularios HTML, ya que permite al usuario introducir diferentes tipos de datos. El comportamiento y la apariencia de un `<input>` dependen del valor del atributo `type` , que define el tipo de dato que puede recibir.

Tipos comunes de `<input>` :

- `text` : entrada de texto de una sola línea.

```
<input type="text" name="nombre" placeholder="Introduce tu nombre">
```

- `password` : campo de entrada para contraseñas, donde los caracteres introducidos se muestran enmascarados.

```
<input type="password" name="password" placeholder="Introduce tu contraseña">
```

- `email` : campo de entrada para correos electrónicos, que puede validar el formato del correo automáticamente.

```
<input type="email" name="correo" placeholder="Introduce tu correo">
```

- `number` : campo de entrada para números. Puedes establecer límites mediante los atributos `min` y `max` .

```
<input type="number" name="edad" min="1" max="120">
```

- `radio` : permite seleccionar una opción entre varias. Todos los botones de radio con el mismo `name` están agrupados.

```
<input type="radio" name="genero" value="hombre"> Hombre
<input type="radio" name="genero" value="mujer"> Mujer
```

- `checkbox` : permite seleccionar o deseleccionar una opción,

```
<input type="checkbox" name="terminos" value="acepto"> Acepto los términos y condiciones
```

- `date` : campo de entrada para seleccionar una fecha.

```
<input type="date" name="fecha_nacimiento">
```

- `file` : permite subir archivos desde el sistema del usuario.

```
<input type="file" name="curriculum">
```

- `submit` : botón que envía los datos del formulario al servidor.

```
<input type="submit" value="Enviar">
```

Los navegadores modernos proporcionan validación automática para tipos como `email` , `number` y `url` , lo que facilita la captura de datos válidos.

## Elemento `<button>`

El elemento `<button>` se utiliza para crear **botones interactivos** en un formulario o en otras partes de una página web. A diferencia del elemento `<input type="submit">` , el elemento `<button>` permite incluir contenido HTML adicional dentro de él, como texto, imágenes o iconos.

Existen tres tipos principales de botones que se pueden especificar mediante el atributo `type` :

- `<button type="submit">` : envía el formulario al servidor. Este es el valor por defecto si no se especifica ningún `type`.
- `<button type="reset">` : restablece todos los campos del formulario a sus valores iniciales.
- `<button type="button">` : no tiene un comportamiento por defecto, lo que lo convierte en un botón genérico que puede utilizarse para realizar acciones mediante JavaScript.

Además, se pueden aplicar varios atributos comunes, como `disabled` , `autofocus` , y `name` .

```
<form method="post" action="process.php">
  <button type="submit">Enviar</button>
  <button type="reset">Restablecer</button>
  <button type="button" onclick="alert('Botón genérico')">Haz clic aquí</button>
</form>
```

El contenido de los botones puede incluir texto e incluso imágenes, lo que hace al `<button>` más flexible que los elementos `<input>` de tipo botón:

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

## Elemento `<select>`

El elemento `<select>` se utiliza para crear un **menú desplegable** en un formulario, donde el usuario puede elegir una o más opciones de una lista.

Las opciones dentro de un elemento `<select>` se definen mediante el elemento `<option>`. También se puede agrupar un conjunto de opciones relacionadas usando `<optgroup>`.

- `multiple` (booleano): permite la selección de múltiples opciones si se establece.
- `size`: especifica cuántas opciones se deben mostrar a la vez. Si no se establece, el menú se muestra como un desplegable.

```
<form method="post" action="process.php">
  <label for="colors">Choose a color:</label>
  <select id="colors" name="color">
    <option value="red">Red</option>
    <option value="blue">Blue</option>
    <option value="green">Green</option>
  </select>
  <button type="submit">Submit</button>
</form>
```

En caso de usar el atributo `multiple`, los datos se enviarán al servidor en forma de array. Para gestionar esto correctamente en el servidor, es recomendable añadir corchetes ( `[]` ) al nombre del campo:

```
<select name="hobbies[]" multiple>
  <option value="leer">Leer</option>
  <option value="cocinar">Cocinar</option>
  <option value="programar">Programar</option>
</select>
```

Si el usuario selecciona varias opciones, el servidor recibirá un array con todas las opciones elegidas, como por ejemplo:

```
Array
(
    [hobbies] => Array
        (
            [0] => leer
            [1] => programar
        )
)
```

## Elemento `<optgroup>`

El elemento `<optgroup>` se utiliza dentro de un elemento `<select>` para **agrupar opciones relacionadas**. Esto ayuda a organizar y categorizar las opciones disponibles, facilitando así la selección para los usuarios.

El atributo `label` especifica una etiqueta que describe el grupo de opciones. Esta etiqueta es lo que se mostrará al usuario para identificar el grupo.

```
<select>
  <optgroup label="Frutas">
    <option value="apple">Manzana</option>
    <option value="banana">Plátano</option>
    <option value="orange">Naranja</option>
  </optgroup>
  <optgroup label="Verduras">
```

```

    <option value="carrot">Zanahoria</option>
    <option value="broccoli">Brócoli</option>
    <option value="lettuce">Lechuga</option>
  </optgroup>
</select>

```

## Elemento `<option>`

El elemento `<option>` se utiliza dentro de los elementos `<select>` y `<optgroup>` para definir **las opciones que un usuario puede seleccionar**. Cada `<option>` representa una única elección dentro de un conjunto.

- `value` especifica el valor que se enviará al servidor cuando se selecciona la opción.
- `label` permite especificar una etiqueta alternativa que se mostrará en la lista desplegable. Esto puede ser útil si deseas que el usuario vea un texto diferente al valor que se enviará.
- `selected` (booleano) indica que la opción debe estar seleccionada por defecto cuando se carga el formulario.

```

<select>
  <option value="apple">Manzana</option>
  <option value="banana" selected>Plátano</option>
  <option value="orange">Naranja</option>
  <option value="grape" disabled>Uva (no disponible)</option>
</select>

```

## Elemento `<datalist>`

El elemento `<datalist>` se utiliza para proporcionar **una lista de opciones predefinidas** para un campo de entrada. Este elemento mejora la experiencia del usuario al permitirle seleccionar de una lista sugerida mientras escribe, lo que facilita la entrada de datos y puede ayudar a evitar errores de escritura.

Un `<datalist>` debe estar asociado a un elemento `<input>` mediante el atributo `list`. Cuando el usuario comienza a escribir en el campo de entrada, el navegador muestra las opciones disponibles en la lista asociada, permitiendo al usuario elegir una de ellas.

El atributo `label` se utiliza para especificar una etiqueta que describa el grupo de opciones. Esta etiqueta es lo que se mostrará al usuario para identificar el grupo.

```

<form>
  <label for="fruits">Choose a fruit:</label>
  <input type="text" id="fruits" list="fruit-list">
  <datalist id="fruit-list">
    <option value="Apple">
    <option value="Banana">
    <option value="Cherry">
    <option value="Date">
    <option value="Grape">
    <option value="Kiwi">
  </datalist>
</form>

```

En este ejemplo, el usuario puede comenzar a escribir en el campo de texto y se le mostrarán sugerencias de frutas basadas en las opciones definidas en el `<datalist>`. Esto hace que la entrada de datos sea más rápida y sencilla.

El contenido de un elemento `<datalist>` puede ser generado dinámicamente desde el backend. Esto es especialmente útil si deseas que la lista de opciones dependa de datos almacenados en una base de datos o de la lógica de tu aplicación.

## Elemento `<textarea>`

El elemento `<textarea>` se utiliza en los formularios HTML para permitir **la entrada de texto en varias líneas**. A diferencia del elemento `<input>`, que se usa para una sola línea de texto, `<textarea>` está diseñado para capturar grandes bloques de texto.

- `cols` : define el número de columnas visibles (el ancho) del área de texto.
- `rows` : define el número de filas visibles (la altura) del área de texto.
- `maxLength` : establece el número máximo de caracteres que se pueden ingresar.
- `wrap` : define cómo se manejarán las líneas largas:
  - `soft` : las líneas no se dividen al enviar el formulario.
  - `hard` : las líneas se dividen y los saltos de línea se envían como parte del valor.

```
<form>
  <label for="comments">Comentarios:</label>
  <textarea id="comments" name="comments" rows="4" cols="50" placeholder="Escriba aquí sus comentarios..."></textarea>
</form>
```

A diferencia del campo `<input>`, el contenido del campo `<textarea>` va entre las etiquetas de apertura y cierre del elemento, lo que permite mostrar contenido predefinido.

## Elemento `<output>`

El elemento `<output>` se utiliza para **mostrar el resultado** de un cálculo o la salida de un formulario. Es particularmente útil en situaciones donde los usuarios interactúan con formularios y se requiere mostrar resultados dinámicamente, como cálculos de totales, resultados de evaluaciones, o cualquier dato que dependa de la entrada del usuario.

Este elemento se beneficia de su semántica, ya que indica claramente que el contenido mostrado es el resultado de una acción o cálculo, mejorando la accesibilidad y la comprensión del contenido.

El atributo `for` especifica los elementos de formulario cuyos valores se utilizan para calcular el resultado mostrado en el `<output>`. Este atributo permite asociar el `<output>` a uno o más controles de formulario.

```
<form oninput="result.value=parseInt(a.value) + parseInt(b.value)">
  <label for="a">A: <input type="number" id="a" value="0"></label>
  <label for="b">B: <input type="number" id="b" value="0"></label>
  <output name="result" for="a b">0</output>
</form>
```

En este ejemplo, el `<output>` muestra la suma de los valores introducidos en los campos A y B. A medida que el usuario ingresa datos, el resultado se actualiza dinámicamente, proporcionando una experiencia interactiva.

## Elemento `<progress>`

El elemento `<progress>` se utiliza para mostrar **el progreso de una tarea en curso**. Es ideal para representar visualmente el avance de acciones que requieren tiempo, como la carga de archivos, la descarga de datos, o el progreso de una operación en una aplicación web.

Este elemento es especialmente útil porque proporciona una indicación clara y comprensible del estado de una tarea, mejorando la experiencia del usuario.

- `value` especifica el progreso actual de la tarea. Este valor debe estar entre 0 y el valor máximo.
- `max` define el valor máximo de progreso. Si no se especifica, el valor por defecto es 1.

```
<form>
  <label for="fileUpload">Uploading file:</label>
  <progress id="fileUpload" value="0.6" max="1"></progress>
</form>
```

En este ejemplo, el `<progress>` muestra que la tarea de carga de un archivo está al 60% de progreso. Esto permite a los usuarios ver de manera visual cuánto queda para completar la operación.

## Elemento `<meter>`

El elemento `<meter>` se utiliza para representar **un valor escalar dentro de un rango conocido**, como puede ser una medida de rendimiento o progreso. Es comúnmente utilizado para mostrar datos como el nivel de batería, el progreso de una tarea, o el nivel de un recurso.

Este elemento proporciona una forma visual de representar datos que pueden ser interpretados fácilmente por los usuarios, mejorando así la experiencia de interacción con el formulario.

- `value` : especifica el valor actual que se está midiendo.
- `min` : establece el valor mínimo que puede representar el `<meter>` . Si no se especifica, el valor por defecto es 0.
- `max` : establece el valor máximo que puede representar el `<meter>` . Si no se especifica, el valor por defecto es 100.
- `low` : indica el valor que se considera bajo dentro del rango. Cuando el valor actual es inferior a este, el indicador puede mostrarse en un color diferente.
- `high` : indica el valor que se considera alto dentro del rango. Similar al atributo low, puede cambiar la representación visual si el valor actual supera este umbral.
- `optimum` : define el valor óptimo dentro del rango, que podría marcarse de manera especial en la visualización.

```
<form>
  <label for="battery">Battery Level:</label>
  <meter id="battery" value="70" min="0" max="100" low="20" high="80" optimum="50"></meter>
</form>
```

En este ejemplo, el `<meter>` representa un nivel de batería del 70%, con indicadores de bajo (20%), alto (80%) y óptimo (50%). Esto permite al usuario entender rápidamente la condición del nivel de batería.

## Recursos multimedia embebidos

Los [recursos multimedia](#) embebidos, que incluyen tanto audio como vídeo, son elementos clave para enriquecer la experiencia del usuario en la web.

Para poder reproducir archivos multimedia, se requiere un **codec**, que es un acrónimo de **C**Odificador-**D**Ecodificador. Algunos ejemplos de códecs son VP9, MP3, AAC, y H.265. Una vez codificados, los archivos multimedia deben ser "empaquetados" en un formato de transporte como .ogg, .mp4 o .webm.

## Elemento `<source>`



El elemento `<source>` se puede utilizar con los elementos `<img>`, `<video>` y `<audio>`.

Este elemento permite **definir diferentes fuentes de datos** para un mismo recurso multimedia. El navegador elegirá automáticamente el origen que mejor se ajuste a sus capacidades y preferencias del usuario.

## Elemento `<video>`

El elemento `<video>` permite insertar archivos de vídeo en un documento web.

Aunque se puede utilizar el atributo `src` para indicar el archivo de vídeo a utilizar, es recomendable emplear la etiqueta `<source>` para proporcionar múltiples formatos y garantizar la compatibilidad en diferentes navegadores.

También es posible incluir un mensaje de texto que se mostrará si el navegador no admite la reproducción de vídeos.

```
<video controls poster="intro.jpg">
  <source src="clipA.mp4">
  <source src="clipA.webm">
  <p>El navegador no tiene soporte para reproducir estos clips</p>
</video>
```

Atributos importantes:

- `autoplay` (booleano): reproduce automáticamente el vídeo al cargar la página.
- `controls` (booleano): muestra controles de reproducción nativos que permiten al usuario pausar, avanzar y retroceder.
- `poster`: especifica una imagen que se mostrará antes de que se inicie la reproducción del vídeo.
- `width` y/o `height`: permiten definir las dimensiones del vídeo.
- `loop` (booleano): hace que el vídeo se reproduzca en bucle.
- `muted` (booleano): desactiva el sonido del vídeo.

Para optimizar la experiencia de reproducción, se recomienda precargar el vídeo utilizando el atributo `preload` con los siguientes valores:

- `preload="auto"`: descarga automáticamente los datos necesarios.
- `preload="metadata"`: descarga solo los metadatos del vídeo (duración, tamaño, etc..).
- `preload="none"`: no precarga el vídeo.

## Elemento `<audio>`

El elemento `<audio>` permite insertar archivos de audio en un documento web.

Al igual que con el vídeo, se recomienda utilizar la etiqueta `<source>` para especificar diferentes formatos y ofrecer compatibilidad con múltiples navegadores:

```
<audio controls>
  <source src="audioA.mp3">
  <source src="audioA.webm">
  <p>El navegador no soporta esta etiqueta</p>
</audio>
```

Atributos importantes:

- `controls` : muestra los controles para la reproducción de audio.
- `autoplay` : reproduce el audio automáticamente al cargar la página.
- `loop` : reproduce el audio en bucle.
- `muted` : Desactiva el sonido del audio.

## Validación de errores en HTML

---

En los lenguajes de marcas como HTML, los navegadores suelen ser más permisivos. Cuando encuentran un error, intentan «deducir» la intención del desarrollador y continúan cargando el documento, lo que puede llevar a comportamientos inesperados.

En un código HTML podemos encontrar varios tipos de problemas:

- **Sintaxis**: se refiere a errores en la escritura del código que pueden impedir que ciertos elementos se muestren correctamente.
- **Accesibilidad**: aunque el código puede ser técnicamente correcto, podría presentar problemas que dificulten su uso en determinados dispositivos o por personas con discapacidad visual.
- **Posicionamiento (SEO)**: un código bien escrito no garantiza un buen posicionamiento en buscadores. Problemas en la estructura del HTML pueden afectar la forma en que los motores de búsqueda indexan la página.
- **Rendimiento**: el código puede no tener errores sintácticos, pero una estructura ineficiente puede ralentizar la carga del navegador o provocar retrasos en las tareas.
- **Usabilidad**: aunque el código sea correcto, la disposición de los elementos puede frustrar a los usuarios, afectando su experiencia en la web.



Para asegurarnos de que nuestro código HTML esté correctamente escrito y libre de errores, podemos utilizar un **Validador HTML**. Esta herramienta analiza nuestro código y proporciona un informe sobre el número de errores encontrados, junto con descripciones que facilitan su corrección.

El proceso de validación se puede realizar a través de la herramienta oficial [HTML Validator de W3C](#) o mediante [plugins para el IDE o paquetes NPM](#) que permiten una validación integrada en el flujo de trabajo del desarrollo.



---



## Enlaces

### HTML5

-  [HTML Specification](#)
- [W3C standards and drafts](#)
-  [Awesome - A curated list of awesome HTML5 resources](#)

### HTML5 - Learning

-   [htmlreference.io - A free guide to all HTML elements and attributes.](#)
- <https://developer.mozilla.org/en-US/docs/Web/HTML>
- ES <https://lenguajehtml.com/html/>

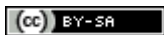
- <https://w3schools.com/html/>
- <https://cheatsheets.zip/>
- <https://internetingishard.netlify.app/html-and-css/>
- <https://theodinproject.com/>
- <https://roadmap.sh/frontend>
- <https://html.com/>
- <https://cheatsheets.shcodes.io/html>
- <https://overapi.com/html>
- <https://htmlcheatsheet.com/>
- <https://devhints.io/>
-  <https://goalkicker.com/HTML5Book/>
-  <https://goalkicker.com/HTML5CanvasBook/>
- <https://web.dev/learn/html>

## HTML5 - Other

- "Can I use"
- HTML5 Please - Look up HTML5, CSS3, etc features, know if they are ready for use
- HTML5 Boilerplate - The web's most popular front-end template
- Modernizr: the feature detection library for HTML5/CSS3
- htmx is a library that allows you to access modern browser features directly from HTML

## Licencia

---



Esta obra está bajo una [licencia de Creative Commons Reconocimiento-Compartir Igual 4.0 Internacional](https://creativecommons.org/licenses/by-sa/4.0/).