

# hello\_

COLECCIÓN FRONTEND

# HTML5 para Dummies

(y para quienes quieren dejar de copiar y  
pegar sin entender)



porque todo gran  
desarrollador  
empezó con un  
`<h1>`

Aprende a poner orden con  
`<header>` y `<footer>`

Crearás webs desde el primer  
minuto... bueno mejor 15min.

Haz que Google te quiera (o al  
menos te entienda)

**@mihifidem**

Creador digital, autor espontáneo  
y domador de etiquetas rebeldes.

# HTML5



# Tabla de contenidos

## HTML5

<b>1</b>	<b>Estructura</b>	<b>2</b>
	Tipo de documento	2
	Elementos estructurales	3
	Atributos globales	18
<b>2</b>	<b>Contenido</b>	<b>20</b>
	Texto	20
	Enlaces	27
	Imágenes	33
	Listados	36
	Tablas	40
	Atributos globales	44
<b>3</b>	<b>Formularios</b>	<b>46</b>
	Definición	46
	Elementos	48
	Enviando el formulario	66
	Atributos globales	70
	Listado de ejercicios	79

## 1. Estructura

El código HTML sigue siendo la parte fundamental para la creación de sitios web y aplicaciones. Esta estructura define el espacio donde el contenido estático y dinámico es posicionado, siendo la plataforma básica para toda aplicación. Para crear un sitio o una aplicación, se debe comenzar programando el código HTML para definir la estructura de las páginas.



**IMPORTANTE:** los documentos HTML son archivos de texto que se pueden crear con cualquier editor de texto o los editores profesionales indicados en el Tema 1. La mayoría de estos editores ofrecen herramientas para ayudarle a escribir sus documentos, pero no controlan la validez del código. Si omite una etiqueta o se olvida de escribir una parte del código, el editor no le advierte sobre el error cometido. Para controlar el código de sus documentos, puede usar herramientas de validación en línea. La más popular para documentos HTML se encuentra disponible en <http://validator.w3.org>.

### Tipo de documento

La declaración `<!DOCTYPE>` es necesaria para indicar el tipo de archivo que se está procesando al navegador. Esta etiqueta se agrega al comienzo del documento para ayudar al navegador a generar la página web correctamente. En el caso de documentos programados con HTML5, la declaración debe contener el atributo `html`.

---

```
<!DOCTYPE html>
```

---

#### *Listado 2-1: Incluyendo la declaración `<!DOCTYPE>`*



**Ejercicio:** abra VSC y cree un nuevo archivo llamado `index.html` para probar los códigos de este tema



**IMPORTANTE:** la línea con la declaración `<!DOCTYPE>` debe ser la primera línea de su documento, sin ningún espacio o código previo. Esto activa el modo estándar y obliga a los navegadores a interpretar HTML5 cuando es posible o ignorarlo en caso contrario.



**TIPS:** Para construir una estructura completa de un fichero HTML, pulsar `!+TAB`



**Lo básico:** algunos de los elementos y atributos introducidos en HTML5 no están disponibles en viejos navegadores como Internet Explorer. Para saber qué navegadores implementan estos elementos u otras funciones incorporadas por HTML5, visite [www.caniuse.com](http://www.caniuse.com). Este sitio web ofrece una lista de todos los elementos, atributos, propiedades CSS, y códigos JavaScript disponibles en HTML5, junto con los navegadores que los admiten.

## Elementos estructurales

El lenguaje HTML consta de una estructura de árbol que se basa en el elemento `<html>` como su raíz. Existen diversos elementos para definir la estructura de la página web, permitiendo al navegador mostrar correctamente el contenido en la pantalla. Estos elementos contienen secciones generales, así como contenido más específico.

- **`<html>`**—Este elemento delimita el código HTML. Puede incluir el atributo `lang` para definir el idioma del contenido del documento.
- **`<head>`**—Este elemento se usa para definir la información necesaria para configurar la página web, como el título, el tipo de codificación de caracteres y los archivos externos requeridos por el documento.
- **`<body>`**—Este elemento delimita el contenido del documento (la parte visible de la página).

---

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

</html>
```

---

**Listado 2-2:** Incluyendo el elemento `<html>`



**Lo básico:** existen varios valores disponibles para el atributo `lang`, incluidos **en** para inglés, **es** para español, **fr** para francés, entre otros. Para obtener una lista completa, visite [https://en.wikipedia.org/wiki/List\\_of\\_ISO\\_639-1\\_codes](https://en.wikipedia.org/wiki/List_of_ISO_639-1_codes).

El código HTML insertado entre las etiquetas `<html>` se tiene que dividir en dos secciones principales: la cabecera y el cuerpo. Por supuesto, la cabecera va primero y, al igual que el resto de los elementos estructurales, está compuesta por etiquetas de apertura y cierre.

---

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

</head>
</html>
```

---

**Listado 2-3:** Incluyendo el elemento `<head>`

Entre las etiquetas `<head>` debemos definir el título de la página web, declarar el tipo de codificación de caracteres, facilitar información general acerca del documento, e incorporar los archivos externos con estilos y códigos necesarios para generar la página. Excepto por el título e iconos, el resto de la información insertada en medio de estas etiquetas no es visible para el usuario.

La otra sección que forma parte de la organización principal de un documento HTML es el cuerpo, la parte visible del documento que se especifica con el elemento `<body>`.

---

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

</head>
<body>

</body>
</html>
```

---

**Listado 2-4:** Incluyendo el elemento `<body>`



**Lo básico:** como ya mencionamos, la estructura HTML puede describirse como un árbol, con el elemento `<html>` como su raíz, pero otra forma de definir la relación entre los elementos es describirlos como padres, hijos o hermanos, de acuerdo a sus posiciones en la estructura. Por ejemplo, en un documento HTML típico, el elemento `<body>` es hijo del elemento `<html>` y hermano del elemento `<head>`. Ambos, `<body>` y `<head>`, tienen al elemento `<html>` como su padre.

La estructura básica ya está lista. Ahora tenemos que construir la página, comenzando por la definición de la cabecera. La cabecera incluye toda la información y los recursos necesarios para generar la página. Los siguientes son los elementos disponibles para este propósito.

- **<title>**—Este elemento define el título de la página.
- **<base>**—Este elemento define la URL usada por el navegador para establecer la ubicación real de las URL relativas. El elemento debe incluir el atributo href para declarar la URL base. Cuando se declara este elemento, en lugar de la URL actual, el navegador usa la URL asignada al atributo href para completar las URL relativas.
- **<meta>**—Este elemento representa metadatos asociados con el documento, como la descripción del documento, palabras claves, el tipo de codificación de caracteres, etc. El elemento puede incluir los atributos name para describir el tipo de metadata, content para especificar el valor, y charset para declarar el tipo de codificación de caracteres a utilizar para procesar el contenido.
- **<link>**—Este elemento especifica la relación entre el documento y un recurso externo (generalmente usado para cargar archivos CSS). El elemento puede incluir los atributos href para declarar la ubicación del recurso, rel para definir el tipo de relación, media para especificar el medio al que el recurso está asociado (pantalla, impresora, etc.), y type y sizes para declarar el tipo de recurso y su tamaño (usado a menudo para cargar iconos).
- **<style>**—Este elemento se usa para declarar estilos CSS dentro del documento (estudiado en el Capítulo 3).
- **<script>**—Este elemento se usa para cargar o declarar código JavaScript (estudiado en el Capítulo 6).

Lo primero que tenemos que hacer cuando declaramos la cabecera del documento es especificar el título de la página con el elemento **<title>**. Este texto es el que muestran los navegadores en la parte superior de la ventana, y es lo que los usuarios ven cuando buscan información en nuestro sitio web por medio de motores de búsqueda como Google o Yahoo.

---

```
<!DOCTYPE html>
<html lang="es">
<head>
  <title>Este texto es el título del documento</title>
</head>
<body>

</body>
</html>
```

---

**Listado 2-5:** Incluyendo el elemento **<title>**



**Ejercicio 2.1:**reemplace el código en su archivo `index.html` por el código del Listado 2-5 y abra el documento en su navegador (para abrirlo, puede hacer doble clic en el archivo o seleccionar la opción **Abrir Archivo** desde el menú **Archivos** en su navegador). Debería ver el texto especificado entre las etiquetas `<title>` en la parte superior de la ventana.



**Lo básico:** el elemento `<title>` en el ejemplo del Listado 2-5 se ha desplazado hacia la derecha. El espacio en blanco en el lado izquierdo se usa para ayudar al desarrollador a visualizar la posición del elemento dentro de la jerarquía del documento. Este espacio se genera automáticamente por editores como Atom, pero puede hacerlo usted mismo cuando lo necesite pulsando la tecla Tab (Tabulador) en su teclado (los navegadores ignoran los espacios en blanco y los saltos de líneas que se encuentran fuera de los elementos).

Además del título, también tenemos que declarar los metadatos del documento. Los metadatos incluyen información acerca de la página que los navegadores, y también los motores de búsqueda, utilizan para generar y clasificar la página web. Los valores se declaran con el elemento `<meta>`. Este elemento incluye varios atributos, pero cuáles usemos dependerá del tipo de información que queremos declarar. Por ejemplo, el valor más importante es el que define la tabla de caracteres a utilizar para presentar el texto en pantalla, el cual se declara con el atributo `charset`.

---

```
<!DOCTYPE html>
<html lang="es">
<head>
  <title>Este texto es el título del documento</title>
  <meta charset="utf-8">
</head>
<body>

</body>
</html>
```

---

#### **Listado 2-6:** Incluyendo el elemento `<meta>`



**Lo básico:** el ejemplo del Listado 2-6 define el grupo de caracteres como `utf-8`, que es el que se recomienda debido a que incluye todos los caracteres utilizados en la mayoría de los idiomas, pero existen otros disponibles. Para más información, visite nuestro sitio web y siga los enlaces de este capítulo.



Se pueden incluir múltiples elementos `<meta>` para declarar información adicional. Por ejemplo, dos datos que los navegadores pueden considerar a la hora de procesar nuestros documentos son la descripción de la página y las palabras claves que identifican su contenido. Estos elementos `<meta>` requieren el atributo `name` con los valores "description" y "keywords", y el atributo `content` con el texto que queremos asignar como descripción y palabras clave (las palabras clave se deben separar por comas).

---

```
<!DOCTYPE html>
<html lang="es">
<head>
  <title>Este texto es el título del documento</title>
  <meta charset="utf-8">
  <meta name="description" content="Este es un documento HTML5">
  <meta name="keywords" content="HTML, CSS, JavaScript">
</head>
<body>

</body>
</html>
```

---

**Listado 2-7:** Agregando información adicional con el elemento `<meta>`

Otro elemento importante de la cabecera del documento es `<link>`. Este elemento se usa para incorporar al documento estilos, códigos, imágenes o iconos desde archivos externos. Por ejemplo, algunos navegadores muestran un icono en la parte superior de la ventana junto con el título de la página. Para cargar este icono, tenemos que incluir un elemento `<link>` con el atributo `rel` definido como `icon`, el atributo `href` con la ubicación del archivo que contiene el icono, el atributo `type` para especificar el formato con el que se ha creado el icono, y el atributo `sizes` con el ancho y la altura del icono separados por la letra `x`.

---

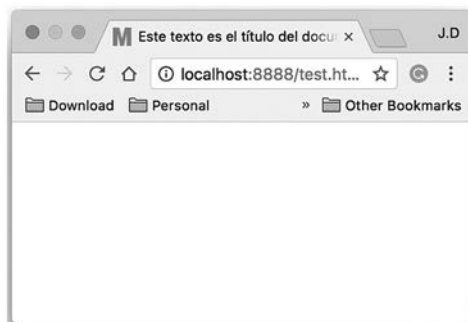
```
<!DOCTYPE html>
<html lang="es">
<head>
  <title>Este texto es el título del documento</title>
  <meta charset="utf-8">
  <meta name="description" content="Este es un documento HTML5">
  <meta name="keywords" content="HTML, CSS, JavaScript">
  <link rel="icon" href="imagenes/favicon.png" type="image/png"
  sizes="16x16">
</head>
<body>

</body>
</html>
```

---

**Listado 2-8:** Incluyendo el icono del documento

El navegador tiene poco espacio para mostrar el icono, por lo tanto el tamaño típico de esta imagen es de unos 16 píxeles por 16 píxeles. La Figura 2-1 muestra cómo se ve la ventana cuando abrimos un documento que contiene un icono (en este caso, se muestra una imagen con la letra M en el lado izquierdo del título).



**Figura 2-1:** El icono del documento en el navegador



**Ejercicio 2.2 :** actualice el código en su archivo `index.html` con el ejemplo del Listado 2-8. El icono se carga desde el archivo `favicon.png` que debe copiar dentro del directorio de su proyecto. Puede descargar este archivo desde un sitio web o usar el suyo.



**Lo básico:** el valor asignado al atributo `type` del elemento `<link>` debe ser especificado como un tipo MIME. Todo archivo tiene un tipo MIME asociado para indicar al sistema el formato de su contenido. Por ejemplo, el tipo MIME de un archivo HTML es `text/html`. Existe un tipo MIME para cada tipo de archivo disponible, que incluye `image/jpeg` e `image/png` para imágenes JPEG y PNG. Para obtener una lista completa, visite nuestro sitio web y siga los enlaces de este capítulo.



**Ejercicio 2.3 :** Analiza el fichero `index_2_3.html` y detecta todos los errores estructurales.

El elemento `<link>` se usa comúnmente para cargar archivos CSS con los estilos necesarios para generar la página web. Por ejemplo, el siguiente documento carga el archivo `misestilos.css`. Después de cargar el archivo, todos los estilos declarados en su interior se aplican a los elementos del documento. En este caso, solo necesitamos incluir el atributo `rel` para declarar el tipo de recurso (para hojas de estilo CSS debemos asignar el valor "stylesheet") y el atributo `href` con la URL que determina la ubicación del archivo (estudiaremos cómo crear esta clase de archivos y definir estilos CSS en el Capítulo 3).

---

```
<!DOCTYPE html>
<html lang="es">
<head>
  <title>Este texto es el título del documento</title>
  <meta charset="utf-8">
  <meta name="description" content="Este es un documento HTML5">
  <meta name="keywords" content="HTML, CSS, JavaScript">
  <link rel="stylesheet" href="misestilos.css">
</head>
<body>

</body>
</html>
```

---

**Listado 2-9:** Cargando un archivo CSS con el elemento `<link>`

Con la cabecera lista, es hora de construir el cuerpo. Esta estructura (el código entre las etiquetas `<body>`) es la encargada de generar la parte visible de nuestro documento (la página web).

HTML siempre ha permitido diferentes formas de construir y organizar la información en el documento. Con la introducción de CSS, los elementos `<table>` (tabla) ya no resultaban prácticos, de ahí la implementación de `<div>` (división). Sin embargo, `<div>` no ofrecía información acerca de la parte del documento que representaba y se podían insertar cualquier cosa entre sus etiquetas. HTML5 ha introducido nuevos elementos con nombres descriptivos para identificar cada parte del documento y que informen al navegador sobre su propósito. Esto facilita la creación del documento y ayuda a los desarrolladores a definir su estructura.

- **<div>**—Este elemento define una división genérica. Se usa cuando no se puede aplicar ningún otro elemento.
- **<main>**—Este elemento define una división que contiene el contenido principal del documento (el contenido que representa el tema central de la página).
- **<nav>**—Este elemento define una división que contiene ayuda para la navegación, como el menú principal de la página o bloques de enlaces necesarios para navegar en el sitio web.
- **<section>**—Este elemento define una sección genérica. Se usa frecuentemente para separar contenido temático, o para generar columnas o bloques que ayudan a organizar el contenido principal.
- **<aside>**—Este elemento define una división que contiene información relacionada con el contenido principal pero que no es parte del mismo, como referencias a artículos o enlaces que apuntan a publicaciones anteriores.
- **<article>**—Este elemento representa un artículo independiente, como un mensaje de foro, el artículo de una revista, una entrada de un blog, un comentario, etc.
- **<header>**—Este elemento define la cabecera del cuerpo o de secciones dentro del cuerpo.
- **<footer>**—Este elemento define el pie del cuerpo o de secciones dentro del cuerpo.

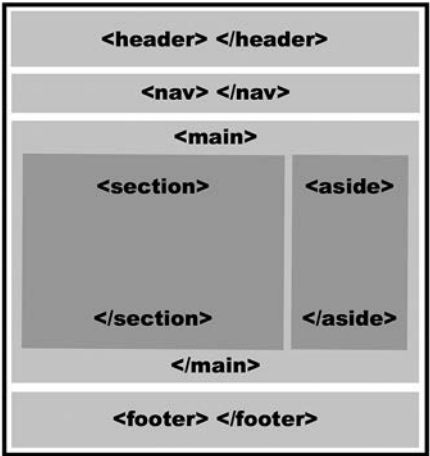
Estos elementos han sido definidos con el propósito de representar secciones específicas de una página web. Aunque son flexibles y se pueden implementar en diferentes partes del diseño, todos siguen un patrón que se encuentra comúnmente en la mayoría de los sitios web. La Figura 2-2, a continuación, ilustra este tipo de diseño.



*Figura 2-2: Representación visual de un diseño web tradicional*

A pesar de que cada desarrollador crea sus propios diseños, en general podremos describir todo sitio web considerando estas secciones. En la barra superior, descrita como cabecera en la Figura 2-2, ubicamos el logo, el nombre del sitio, los subtítulos y una descripción breve de nuestro sitio o página web. En la barra de navegación situada debajo es donde la mayoría de los desarrolladores ofrecen un menú o una lista de enlaces para navegar en el sitio. El contenido relevante de la página se ubica en el medio del diseño, donde generalmente encontramos artículos o noticias, y también enlaces a documentos relacionados o recursos. En el ejemplo de la Figura 2-2, esta sección se ha dividido en dos columnas, información principal y barra lateral, pero los diseñadores la adaptan a sus necesidades insertando columnas adicionales o dividiendo las columnas en bloques más pequeños. En la parte inferior de un diseño tradicional, nos encontramos con otra barra llamada barra institucional. La llamamos de este modo porque en este área es donde mostramos información general acerca del sitio web, el autor, la compañía, los enlaces relacionados con reglas de uso, términos y condiciones, el mapa del sitio, etc.

Como mencionamos anteriormente, los elementos de HTML5 se han diseñado siguiendo este patrón. En la Figura 2-3 aplicamos los elementos introducidos anteriormente para definir el diseño de la Figura 2-2.



**Figura 2-3:** Representación de la estructura de un documento usando elementos de HTML5

Los elementos se declaran en el documento en el mismo orden en el que se presentarán en pantalla, desde la parte superior a la inferior y de izquierda a derecha (este orden se puede modificar por medio de estilos CSS, como veremos en el Capítulo 4). El primer elemento de un diseño tradicional es <header>. No debemos confundir este elemento con el elemento <head> utilizado anteriormente para crear la cabecera del documento. Al igual que <head>, el elemento <header> se ha definido para facilitar información introductoria, como títulos o subtítulos, pero no para el documento, sino para el cuerpo o secciones dentro del cuerpo del documento. En el siguiente ejemplo, este elemento se usa para definir el título de la página web.

---

```
<!DOCTYPE html>
<html lang="es">
<head>
  <title>Este texto es el título del documento</title>
  <meta charset="utf-8">
  <meta name="description" content="Este es un documento HTML5">
  <meta name="keywords" content="HTML, CSS, JavaScript">
  <link rel="stylesheet" href="misestilos.css">
</head>
<body>
  <header>
    Este es el título
  </header>
</body>
</html>
```

---

**Listado 2-10:** Incluyendo el elemento <header>

La inserción del elemento <header> representa el comienzo del cuerpo y de la parte visible del documento. De ahora en adelante, podremos ver el resultado de la ejecución del código en la ventana del navegador.



**Ejercicio 2.4** :reemplace el código en su archivo index.html por el código del Listado 2-10 y abra el documento en su navegador. Debería ver el título de la página en la pantalla.

La siguiente sección de nuestro ejemplo es la barra de navegación. Esta barra define una sección con ayuda para la navegación y se representa con el elemento `<nav>`.

---

```
<!DOCTYPE html>
<html lang="es">
<head>
  <title>Este texto es el título del documento</title>
  <meta charset="utf-8">
  <meta name="description" content="Este es un documento HTML5">
  <meta name="keywords" content="HTML, CSS, JavaScript">
  <link rel="stylesheet" href="misestilos.css">
</head>
<body>
  <header>
    Este es el título
  </header>
  <nav>
    Principal | Fotos | Videos | Contacto
  </nav>
</body>
</html>
```

---

**Listado 2-11:** Incluyendo el elemento `<nav>`

Los elementos HTML son flexibles y su uso depende de las necesidades de nuestro sitio web o aplicación. Sin embargo, es importante seguir los estándares establecidos para mantener el código portable y legible. El elemento `<nav>` debe ser utilizado para contener ayuda para la navegación, como el menú principal o bloques de enlaces importantes.

---

```

<!DOCTYPE html>
<html lang="es">
<head>
  <title>Este texto es el título del documento</title>
  <meta charset="utf-8">
  <meta name="description" content="Este es un documento HTML5">
  <meta name="keywords" content="HTML, CSS, JavaScript">
  <link rel="stylesheet" href="misestilos.css">
</head>
<body>
  <header>
    Este es el título
  </header>
  <nav>
    Principal | Fotos | Videos | Contacto
  </nav>
  <main>
    <section>
      Artículos
    </section>
    <aside>
      Cita del artículo uno
      Cita del artículo dos
    </aside>
  </main>
</body>
</html>

```

---

**Listado 2-12:** Organizando el contenido principal



**IMPORTANTE:** los elementos que representan cada sección del documento se listan en el código uno encima del otro, pero en la página web algunas de estas secciones se muestran una al lado de la otra (por ejemplo, las columnas creadas por las secciones **Información principal** y **Barra lateral** se muestran en una misma línea en la página web). Si abre el documento del Listado 2-12 en su navegador, verá que los textos se muestran uno por línea. Esto se debe a que HTML5 delega la tarea de presentar el documento a CSS. Para mostrar las secciones en el lugar correcto, debemos asignar estilos CSS a cada elemento del documento. Estudiaremos CSS en los Capítulos 3 y 4.

El diseño considerado anteriormente (Figura 2-2) es el más común de todos y representa la estructura básica de la mayoría de los sitios web que encontramos hoy en día, pero es también una muestra de cómo se muestra el contenido importante de una página web en pantalla.



Como los artículos de un diario, las páginas web generalmente presentan la información dividida en secciones que comparten características similares. El elemento `<article>` nos permite identificar cada una de estas partes. En el siguiente ejemplo, implementamos este elemento para representar las publicaciones que queremos mostrar en la sección principal de nuestra página web.

---

```
<!DOCTYPE html>
<html lang="es">
<head>
  <title>Este texto es el título del documento</title>
  <meta charset="utf-8">
  <meta name="description" content="Este es un documento HTML5">
  <meta name="keywords" content="HTML, CSS, JavaScript">
  <link rel="stylesheet" href="misestilos.css">
</head>
<body>
  <header>
    Este es el título
  </header>
  <nav>
    Principal | Fotos | Videos | Contacto
  </nav>
  <main>
    <section>
      <article>
        Este es el texto de mi primer artículo
      </article>
      <article>
        Este es el texto de mi segundo artículo
      </article>
    </section>
    <aside>
      Cita del artículo uno
      Cita del artículo dos
    </aside>
  </main>
</body>
</html>
```

---

**Listado 2-13:** Incluyendo el elemento `<article>`

En este punto, ya contamos con la cabecera y el cuerpo del documento, secciones con ayuda para la navegación y el contenido, e información adicional a un lado de la página. Lo único que nos queda por hacer es cerrar el diseño y finalizar el cuerpo del documento. Con este fin, HTML ofrece el elemento `<footer>`.

---

```
<!DOCTYPE html>
<html lang="es">
<head>
  <title>Este texto es el título del documento</title>
  <meta charset="utf-8">
  <meta name="description" content="Este es un documento HTML5">
  <meta name="keywords" content="HTML, CSS, JavaScript">
  <link rel="stylesheet" href="misestilos.css">
</head>
<body>
  <header>
    Este es el título
  </header>
  <nav>
    Principal | Fotos | Videos | Contacto
  </nav>
  <main>
    <section>
      <article>
        Este es el texto de mi primer artículo
      </article>
      <article>
        Este es el texto de mi segundo artículo
      </article>
    </section>
    <aside>
      Cita del artículo uno
      Cita del artículo dos
    </aside>
  </main>
  <footer>
    &copy; Derechos Reservados 2016
  </footer>
</body>
</html>
```

---

**Listado 2-14:** Incluyendo el elemento `<footer>`

En un diseño web tradicional (Figura 2-2), la sección Barra institucional se define con el elemento <footer>. Esto se debe a que la sección representa el final (o pie) de nuestro documento y se usa comúnmente para compartir información general acerca del autor del sitio o la compañía detrás del proyecto, como derechos de autor, términos y condiciones, etc.

El elemento <footer> se usa para representar el final del documento y tiene el objetivo principal ya mencionado, sin embargo, este elemento y el elemento <header> también se pueden utilizar dentro del cuerpo para representar el comienzo y final de una sección.



**Lo básico:** el ejemplo del Listado 2-14 incluye la cadena de caracteres `&copy;` al pie del documento. Sin embargo, cuando se carga el documento, el navegador reemplaza estos caracteres por el carácter de derechos de autor (©). Estas cadenas de caracteres se denominan *entidades (character entities)* y representan caracteres especiales que no se encuentran en el teclado o tienen un significado especial en HTML, como el carácter de derechos de autor (©), el de marca registrada (®) o los paréntesis angulares usados por HTML para definir los elementos (< y >). Cuando necesite incluir en sus textos uno de estos caracteres, debe escribir la entidad en su lugar. Por ejemplo, si quiere incluir los caracteres < y > dentro de un texto, debe representarlos con las cadenas de caracteres `&lt;` y `&gt;`. Otras entidades de uso común son `&amp;` (&), `&quot;` ("), `&apos;` ('), `&pound;` (£), y `&euro;` (€). Para obtener una lista completa, visite nuestro sitio web y siga los enlaces de este capítulo.

## Atributos globales

Aunque la mayoría de los elementos estructurales tienen un propósito implícito que se refleja en sus nombres, esto no significa que se deban usar solo una vez en el mismo documento. Por ejemplo, algunos elementos como `<section>` y `<aside>` se pueden utilizar muchas veces para representar diferentes partes de la estructura, y otros como `<div>` aún son implementados de forma repetida para separar contenido dentro de secciones. Por esta razón, HTML define atributos globales que podemos usar para asignar identificadores personalizados a cada elemento.

- **id**—Este atributo nos permite asignar un identificador único a un elemento.
- **class**—Este atributo asigna el mismo identificador a un grupo de elementos.

El atributo `id` identifica elementos independientes con un valor único, mientras que el valor del atributo `class` se puede duplicar para asociar elementos con características similares. Por ejemplo, si tenemos dos o más elementos `<section>` que necesitamos diferenciar entre sí, podemos asignar el atributo `id` a cada uno con valores que declaren sus propósitos.

---

```
<main>
  <section id="noticias">
    Artículos largos
  </section>
  <section id="noticiaslocales">
    Artículos cortos
  </section>
  <aside>
    Quote from article one
    Quote from article two
  </aside>
</main>
```

---

**Listado 2-15:** Identificando elementos con el atributo `id`

---

```
<main>
  <section>
    <div class="libros">Libro: IT, Stephen King</div>
    <div class="libros">Libro: Carrie, Stephen King</div>
    <div class="libros">Libro: El resplandor, Stephen King</div>
    <div class="libros">Libro: Misery, Stephen King</div>
  </section>
  <aside>
    Cita del artículo uno
    Cita del artículo dos
  </aside>
</main>
```

---

**Listado 2-16:** Identificando elementos con el atributo `class`

---

En el código del Listado 2-16, tenemos un único elemento `<section>` con el que representamos el contenido principal del documento, pero hemos creado varias divisiones con elementos `<div>` para organizar el contenido. Debido a que estos elementos se han identificado con el atributo `class` y el valor "libros", cada vez que accedemos o modificamos elementos referenciando la clase `libros`, todos estos elementos se ven afectados.



**IMPORTANTE:** los valores de los atributos `id` y `class` son usados por algunos elementos HTML para identificar otros elementos y también por reglas CSS y código JavaScript para acceder y modificar elementos específicos en el documento. En próximos capítulos veremos algunos ejemplos prácticos.



**Lo básico:** los valores asignados a estos atributos son arbitrarios. Puede asignarles cualquier valor que desee, siempre y cuando no incluya ningún espacio en blanco (el atributo `class` utiliza espacios para asignar múltiples clases a un mismo elemento). Así mismo, para mantener su sitio web compatible con todos los navegadores, debería usar solo letras y números, siempre comenzar con una letra y evitar caracteres especiales.

## 2. Contenido

Hemos concluido la estructura básica de nuestro sitio web, pero todavía tenemos que trabajar en el contenido. Los elementos HTML estudiados hasta el momento nos ayudan a identificar cada sección del diseño y asignarles un propósito, pero lo que realmente importa en una página web es lo que hay dentro de esas secciones. Debido a que esta información está compuesta por diferentes elementos visuales, como títulos, textos, imágenes y vídeos, entre otros, HTML define varios elementos para representarla.

### Texto

El medio más importante que puede incluir un documento es texto. HTML define varios elementos para determinar el propósito de cada palabra, frase, o párrafo en el documento. El siguiente elemento se usa para representar títulos.

**<h1>**—Este elemento representa un título. El título se declara entre las etiquetas de apertura y cierre. HTML también incluye elementos adicionales para representar subtítulos, hasta seis niveles (**<h2>**, **<h3>**, **<h4>**, **<h5>**, y **<h6>**).

Cada vez que queremos insertar un título o un subtítulo en el documento, tenemos que incluirlo dentro de etiquetas **<h>**. Por ejemplo, el documento que hemos creado en la sección anterior incluye el título de la página. Como este es el título principal, debería representarse con el elemento **<h1>**, tal como ilustra el siguiente ejemplo.

---

```
<header>
  <h1>Este es el título</h1>
</header>
```

---

**Listado 2-17:** Incluyendo el elemento **<h1>**

Los navegadores otorgan estilos por defecto a los elementos **<h>** que incluyen márgenes y diferentes tamaños de letras, dependiendo de la jerarquía (**<h1>** es el de más alta jerarquía y **<h6>** el de menor jerarquía). La Figura 2-4, debajo, muestra cómo se ve el texto dentro de un elemento **<h1>** con los estilos por defecto.

# Este es el título

Principal | Fotos | Videos | Contacto  
Este es el texto de mi primer artículo  
Este es el texto de mi segundo artículo  
Cita del artículo uno Cita del artículo dos  
© Derechos Reservados 2016

**Figura 2-4:** Título representado por un elemento **<h1>** con estilos por defecto



**Ejercicio 2.5** :reemplace el elemento `<header>` en su archivo `index.html` por el código del Listado 2-17. Abra el documento en su navegador. Debería ver algo similar a lo que se muestra en la Figura 2-4.

Los siguientes son los elementos que ofrece HTML para representar el cuerpo del texto.

**`<p>`**—Este elemento representa un párrafo. Por defecto, los navegadores le asignan un margen en la parte superior para separar un párrafo de otro.

**`<pre>`**—Este elemento representa un texto con formato predefinido, como código de programación o un poema que requiere que los espacios asignados a cada carácter y los saltos de línea se muestren como se han declarado originalmente.

**`<span>`**—Este elemento puede contener un párrafo, una frase o una palabra. No aplica ningún estilo al texto pero se usa para asignar estilos personalizados, como veremos en próximos capítulos.

El elemento `<p>` se utiliza ampliamente para representar el cuerpo del texto. Por defecto, los navegadores les asignan estilos que incluyen márgenes y un salto de línea para diferenciar un párrafo de otro. Debido a estas características, también podemos utilizar los elementos

`<p>` para dar formato a líneas de texto, como las citas de nuestro ejemplo.

---

```
<aside>
  <p>Cita del artículo uno</p>
  <p>Cita del artículo dos</p>
</aside>
```

---

**Listado 2-18:** Definiendo líneas de texto con el elemento `<p>`

El Listado 2-18 presenta las citas dentro del elemento `<aside>` de ejemplos anteriores con elementos `<p>`. Ahora, el navegador muestra cada cita en una línea distinta.

## Este es el título

Principal | Fotos | Videos | Contacto  
Este es el texto de mi primer artículo  
Este es el texto de mi segundo artículo

Cita del artículo uno

Cita del artículo dos

© Derechos Reservados 2016

**Figura 2-5:** Líneas de texto definidas con elementos `<p>`

Cuando un párrafo incluye múltiples espacios, el elemento <p> automáticamente reduce ese espacio a solo un carácter e ignora el resto. El elemento también hace algo similar con los saltos de línea. Todo salto de línea introducido en el documento no se considera cuando el texto se muestra en la pantalla. Si queremos que estos espacios y saltos de línea se muestren al usuario, en lugar de usar el elemento <p> tenemos que usar el elemento <pre>.

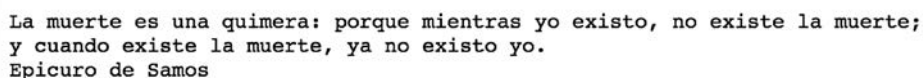
---

```
<article>
  <pre>
    La muerte es una quimera: porque mientras yo existo, no existe la
muerte;
    y cuando existe la muerte, ya no existo yo.
    Epicuro de Samos
  </pre>
</article>
```

---

**Listado 2-19:** *Mostrando texto en su formato original*

El ejemplo del Listado 2-19 define un elemento <article> que contiene una cita de Epicuro de Samos. Como usamos el elemento <pre>, los saltos de línea son considerados por el navegador y las frases se muestran una por línea, tal como se han definido en el código.



```
La muerte es una quimera: porque mientras yo existo, no existe la muerte;
y cuando existe la muerte, ya no existo yo.
Epicuro de Samos
```

**Figura 2-6:** *Texto introducido con un elemento <pre>*

El elemento <pre> se configura por defecto con márgenes y un tipo de letra que respeta el formato asignado al texto original, lo que lo hace apropiado para presentar código de programación y cualquier clase de texto con formato predefinido. En casos como el del ejemplo anterior, donde lo único que necesitamos es incluir saltos de línea dentro del párrafo, podemos usar otros elementos que se han diseñado específicamente con este propósito.

- <br>—Este elemento se usa para insertar saltos de línea.
- <wbr>—Este elemento sugiere la posibilidad de un salto de línea para ayudar al navegador a decidir dónde cortar el texto cuando no hay suficiente espacio para mostrarlo entero.



Estos elementos se insertan dentro del texto para generar saltos de línea. Por ejemplo, podemos escribir el párrafo anterior en una sola línea e insertar elementos `<br>` al final de cada frase para presentarlas en líneas aparte.

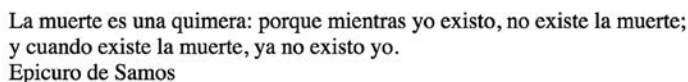
---

```
<article>
  <p>La muerte es una quimera: porque mientras yo existo, no existe la
muerte;<br>y cuando existe la muerte, ya no existo yo.<br>Epicuro de
Samos</p>
</article>
```

---

**Listado 2-20:** Agregando saltos de línea a un párrafo con el elemento `<br>`

A diferencia de los elementos `<p>` y `<pre>`, los elementos `<br>` y `<wbr>` no asignan ningún margen o tipo de letra al texto, por lo que las líneas se muestran como si pertenecieran al mismo párrafo y con el tipo de letra definida por defecto.



La muerte es una quimera: porque mientras yo existo, no existe la muerte;  
y cuando existe la muerte, ya no existo yo.  
Epicuro de Samos

**Figura 2-7:** Saltos de línea generadas por elementos `<br>`

Debido a que no todas las palabras en un texto tienen el mismo énfasis, HTML incluye los siguientes elementos para declarar un significado especial a palabras individuales o frases completas.

**<em>**—Este elemento se usa para indicar énfasis. El texto se muestra por defecto con letra cursiva.

**<strong>**—Este elemento se utiliza para indicar importancia. El texto se muestra por defecto en negrita.

**<i>**—Este elemento representa una voz alternativa o un estado de humor, como un pensamiento, un término técnico, etc. El texto se muestra por defecto con letra cursiva.

**<u>**—Este elemento representa texto no articulado. Por defecto se muestra subrayado.

**<b>**—Este elemento se usa para indicar importancia. Debería ser implementado solo cuando ningún otro elemento es apropiado para la situación. El texto se muestra por defecto en negrita.

Estos elementos se pueden utilizar para resaltar títulos o etiquetas, o para destacar palabras o frases en un párrafo, según muestra el siguiente ejemplo.

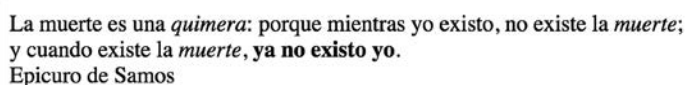
---

```
<article>
  <p>La muerte es una <em>quimera</em>: porque mientras yo existo, no
  existe la <i>muerte</i>;<br>y cuando existe la <i>muerte</i>,
  <strong>ya no existo yo</strong>.<br>Epicuro de Samos</p>
</article>
```

---

**Listado 2-21:** Resaltando texto

A menos que especifiquemos diferentes estilos con CSS, el texto dentro de estos elementos se muestra con los estilos por defecto, como ilustra la Figura 2-8.



La muerte es una *quimera*: porque mientras yo existo, no existe la *muerte*;  
y cuando existe la *muerte*, **ya no existo yo**.  
Epicuro de Samos

**Figura 2-8:** Texto resaltado

La especificidad de elementos estructurales también se manifiesta en algunos de los elementos utilizados para definir el contenido. Por ejemplo, HTML incluye los siguientes elementos para insertar textos que tienen un propósito claramente definido.

**<mark>**—Este elemento resalta texto que es relevante en las circunstancias actuales (porejemplo, términos que busca el usuario).

**<small>**—Este elemento representa letra pequeña, como declaraciones legales, descargos, etc.

**<cite>**—Este elemento representa el autor o título de una obra, como un libro, unapélícula, etc.

**<address>**—Este elemento representa información de contacto. Se implementa con frecuencia dentro de los pies de página para definir la dirección de la empresa o el sitio web.

**<time>**—Este elemento representa una fecha en formato legible para el usuario. Incluye el atributo **datetime** para especificar un valor en formato de ordenador y el atributo **pubdate**, el cual indica que el valor asignado al atributo **datetime** representa la fecha de publicación.

**<code>**—Este elemento representa código de programación. Se usa en conjunto con el elemento **<pre>** para presentar código de programación en el formato original.

**<data>**—Este elemento representa datos genéricos. Puede incluir el atributo **value** para especificar el valor en formato de ordenador (por ejemplo, **<data value="32">Treinta y Dos</data>**).

Como estos elementos representan información específica, normalmente se utilizan paracomplementar el contenido de otros elementos. Por ejemplo, podemos usar el elemento `<time>` para declarar la fecha en la que un artículo se ha publicado y otros elementos como `<mark>` y `<cite>` para otorgarle significado a algunas partes del texto.

---

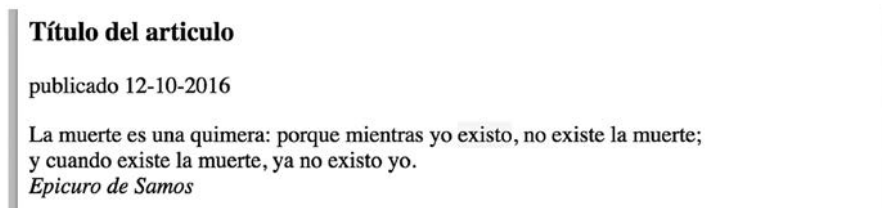
```
<article>
  <header>
    <h1>Título del artículo</h1>
    <time datetime="2016-10-12" pubdate>publicado 12-10-2016</time>
  </header>
  <p>La muerte es una quimera: porque mientras yo <mark>existo</mark>,
no existe la muerte;<br>y cuando existe la muerte, ya no existo
yo.<br><cite>Epicuro de Samos</cite></p>
</article>
```

---

**Listado 2-22:** Complementando el elemento `<article>`

El Listado 2-22 expande el elemento `<article>` utilizado en ejemplos anteriores. Este elemento ahora incluye un elemento `<header>` con el título del artículo y la fecha de publicación, y el texto se ha resaltado con los elementos `<mark>` y `<cite>`.

El elemento `<mark>` resalta partes del texto que originalmente no se consideraban importantes, pero que al momento se han vuelto relevantes (quizás debido a que el usuario realizó una búsqueda con ese texto), y el elemento `<cite>`, en este caso, resalta el nombre del autor. Por defecto, los navegadores asignan estilos al texto dentro del elemento `<mark>` que incluyen un fondo amarillo y muestran el contenido del elemento `<cite>` en cursiva, tal como ilustra la siguiente figura.



**Figura 2-9:** Complementando el elemento `<article>`



**IMPORTANTE:** el valor del atributo `datetime` del elemento `<time>` se debe declarar en formato de ordenador. Este formato requiere la sintaxis `2016-10-12T12:10:45`, donde la T se puede reemplazar por un espacio en blanco y la parte menos significativa se puede ignorar (por ejemplo, `2016-10-12`).



**Lo básico:** el atributo `pubdate` es un atributo booleano. Este tipo de atributos no requieren un valor; representan el valor `true` (verdadero) cuando están presentes o `false` (falso) en caso contrario.


El resto de los elementos mencionados arriba también se combinan con otros elementos para complementar sus contenidos. Por ejemplo, los elementos `<address>` y `<small>` se insertan normalmente dentro de un elemento `<footer>` para resaltar información acerca de la página o una sección.

---

```
<footer>
  <address>Toronto, Canada</address>
  <small>&copy; Derechos Reservados 2016</small>
</footer>
```

---

**Listado 2-23:** Complementando el elemento `<footer>`



Toronto, Canada  
© Derechos Reservados 2016

**Figura 2-10:** Complementando el elemento `<footer>`

El elemento `<code>` también trabaja junto con otros elementos para presentar contenido, pero tiene una relación particular con el elemento `<pre>`. Estos elementos se implementan juntos para presentar código de programación. El elemento `<code>` indica que el contenido es código de programación y el elemento `<pre>` formatea ese contenido para mostrarlo en pantalla como se ha declarado originalmente en el documento (respetando los espacios y los saltos de línea).

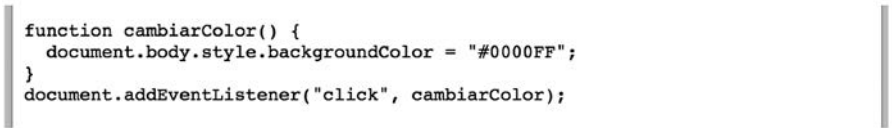
---

```
<article>
  <pre>
    <code>
function cambiarColor() {
  document.body.style.backgroundColor = "#0000FF";
}
document.addEventListener("click", cambiarColor);
    </code>
  </pre>
</article>
```

---

**Listado 2-24:** Presentando código con los elementos `<code>` y `<pre>`

El Listado 2-24 define un elemento `<article>` que muestra código programado en JavaScript. Debido a que este código no se encuentra dentro de un elemento `<script>`, el navegador no lo ejecuta y, debido a que se incluye dentro de un elemento `<pre>`, se presenta con los saltos de línea y los espacios declarados en el documento.



```
function cambiarColor() {
  document.body.style.backgroundColor = "#0000FF";
}
document.addEventListener("click", cambiarColor);
```

**Figura 2-11:** Código de programación presentado con los elementos `<code>` y `<pre>`



**Lo básico:** a veces los nombres de los elementos y su contenido no ofrecen suficiente información al desarrollador para entender el propósito del código. En estas situaciones, HTML nos permite escribir comentarios. Los comentarios son textos que se incluyen en el documento, pero que no son procesados por el navegador. Para agregar un comentario, tenemos que escribir el texto entre las etiquetas `<!--` y `-->`, como en `<!-- Este es un comentario -->`.

## Enlaces

Conectar documentos con otros documentos mediante enlaces es lo que hace posible la Web. Como mencionamos anteriormente, un enlace es contenido asociado a una URL que indica la ubicación de un recurso. Cuando el usuario hace clic en el contenido (texto o imagen), el navegador descarga el recurso. HTML incluye el siguiente elemento para crear enlaces.

**`<a>`**—Este elemento crea un enlace. El texto o la imagen que representa el enlace se incluye entre las etiquetas de apertura y cierre. El elemento incluye el atributo `href` para especificar la URL del enlace.

Los enlaces se pueden crear para conectar el documento actual con otros documentos en el mismo sitio web o en otros sitios. Por ejemplo, podemos enlazar las opciones en el menú de nuestra página web a otros documentos en nuestro servidor.

```
<nav>
  <a href="index.html">Principal</a> |
  <a href="fotos.html">Fotos</a> |
  <a href="videos.html">Videos</a> |
  <a href="contacto.html">Contacto</a>
</nav>
```

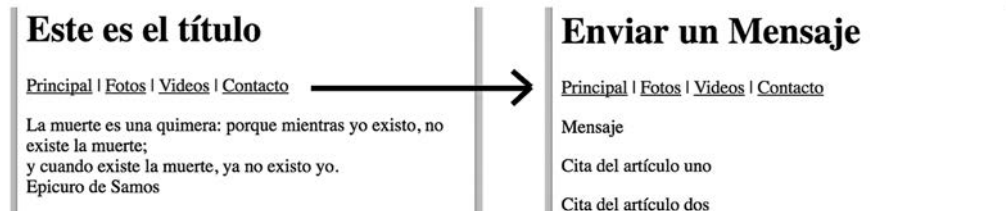
**Listado 2-25:** Enlazando el documento a otros documentos con el elemento `<a>`

El elemento `<nav>` en el Listado 2-25 incluye cuatro elementos `<a>` por cada opción del menú. Los elementos incluyen el atributo `href` para indicar al navegador el documento que tiene que abrir cuando el usuario hace clic en el enlace. Por defecto, los enlaces se muestran subrayados y en color azul (o violeta si el usuario ya ha hecho clic en ellos).

Principal | Fotos | Videos | Contacto

**Figura 2-12:** Hipervínculos

Cuando el usuario hace clic en cualquiera de estos enlaces, el navegador descarga el documento indicado por el atributo href y muestra su contenido en pantalla. Por ejemplo, si hacemos clic en el enlace creado para la opción Contacto en el código del Listado 2-25, el navegador descarga el archivo contacto.html y muestra su contenido en la pantalla, como ilustra la Figura 2-13 (este ejemplo asume que hemos creado un archivo llamado contacto.html).



**Figura 2-13:** Navegando entre documentos



**Ejercicio 2.6 :** cree un nuevo archivo llamado contacto.html. Copie el código HTML del archivo index.html en el archivo contacto.html. Reemplace el elemento `<nav>` en ambos archivos con el código del Listado 2-25. Cambie el título en el elemento `<header>` del archivo contacto.html por el texto «Enviar un Mensaje». Abra el archivo index.html en su navegador y haga clic en la opción **Contacto**. El navegador debería abrir el archivo contacto.html y mostrarlo en pantalla, según ilustra la Figura 2-13.

Los documentos enlazados en el menú del Listado 2-25 pertenecen al mismo sitio web y esa es la razón por la que usamos URL relativas para especificar su ubicación, pero si lo que necesitamos es crear enlaces a documentos que no están almacenados en nuestro servidor, tenemos que usar URL absolutas, como en el siguiente ejemplo.

```
<footer>
  <address>Barcelona, España</address>
  <small>&copy; 2023 <a href="http://www.twooweb.com">Twooweb</a></small>
</footer>
```

**Listado 2-26:** Enlazando el documento a documentos en otros sitios web con el elemento `<a>`

El código en el Listado 2-26 agrega un enlace al pie de página de nuestro ejemplo que apunta al sitio web [www.twooweb.com](http://www.twooweb.com). El enlace trabaja como cualquier otro, pero ahora el navegador tiene la URL completa para acceder al documento (en este caso, el archivo index del sitio web con el domino [www.twooweb.com](http://www.twooweb.com)).

*Barcelona, España*  
© 2023 [Twooweb](http://www.twooweb.com)

**Figura 2-14:** Enlace a un documento externo

El elemento <a> puede incluir el atributo **target** para especificar el destino en el cual el documento será abierto. El valor **\_self** se asigna por defecto, lo que significa que el documento se abre en la misma ubicación que el documento actual (el mismo recuadro o ventana). Otros valores son **\_blank** (el documento se abre en una nueva ventana), **\_parent** (el documento se abre en el recuadro padre), y **\_top** (el documento se abre en la ventana actual).

Como veremos más adelante, los documentos se pueden abrir en recuadros insertados dentro de otros documentos. El valor del atributo target considera esta jerarquía de recuadros, pero debido a que los recuadros no se usan de forma frecuente en sitios web modernos, los dos valores más comunes son **\_self**, para abrir el documento en la misma ventana, y **\_blank**, para abrir el documento en una nueva ventana. El siguiente ejemplo implementa el último valor para acceder al dominio [www.twooweb.com](http://www.twooweb.com) desde una nueva ventana, de modo que el usuario nunca abandona nuestro sitio web.

---

```
<footer>
  <address>Barcelona, España</address>
  <small>&copy; 2023 <a href="http://www.twooweb.com"
target="_blank">Twooweb</a></small>
</footer>
```

---

**Listado 2-27:** *Abriendo un enlace en una nueva ventana*

Además de conectar un documento con otro, los enlaces también se pueden crear hacia otros elementos dentro del mismo documento. Esto es particularmente útil cuando el documento genera una página extensa que el usuario debe desplazar para poder ver todo su contenido. Aprovechando esta característica, podemos crear enlaces hacia diferentes partes de una página. Cuando el usuario quiere ver algo que no es visible al momento, puede hacer clic en estos enlaces y el navegador desplaza la página hasta que el elemento apuntado por el enlace aparece en la pantalla. El elemento que queremos enlazar tiene que ser identificado con el atributo id. Para crear un enlace a un elemento, debemos incluir el valor asignado a este atributo precedido por el carácter #, igual que ilustra el siguiente ejemplo.

---

```

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

<head>
  <title>Este texto es el título del documento</title>
  <meta charset="utf-8">
  <meta name="description" content="Este es un cv web en HTML5">
  <meta name="keywords" content="curriculum web">
</head>

<body>
  <header id="inicio">
    <h1>cv de Oscar Burgos</h1>
  </header>
  <nav>
    Datos personales | Experiencia Laboral | Soft Skills | Hard Skills
  </nav>
  <main>
    <section>
      <h3>Artículo 1</h3>
      <article>Lorem ipsum dolor sit amet, consectetur adipisicing
elit. Consequuntur hic vitae odit iure porro nemo eligendi, autem, iusto
dolore consectetur culpa at placeat aperiam accusantium facere repellendus,
sequi eaque corporis cupiditate quae labore eum! Cumque porro, odio quae
delectus ratione asperiores earum necessitatibus quos itaque ab nihil
blanditiis officiis molestias!</article>

<!-- genera 10 artículos como en anterior. -->

      <p><a href="#inicio">Volver</a></p>
    </section>
  </main>
  <footer>
    &copy; Derechos Reservados 2023
  </footer>
</body>
</html>

```

---

### **Listado 2-28:** *Creando enlaces a elementos en el mismo documento*

En el documento del Listado 2-28, usamos el atributo id con el valor "inicio" para identificar el elemento <header>. Usando este valor y el carácter #, creamos un enlace al final del contenido que lleva al usuario hacia la parte superior de la página. Cuando el usuario hace clic en el enlace, en lugar de abrir un documento, el navegador desplaza la página hasta que el contenido del elemento <header> se vuelve visible.



**Ejercicio 2.7** :actualice el código en su archivo index.html con el código del Listado 2-28. Agregue más párrafos con elementos <p> para generar más contenido dentro del elemento <section>. Abra el documento en su navegador, desplace la página hacia abajo, y haga clic en el enlace **Volver**. Si la cabecera no es visible, el navegador desplazará la página hacia arriba para mostrarla en pantalla.



El elemento `<a>` también se puede usar para crear enlaces a aplicaciones. HTML ofrece las palabras clave `mailto` y `tel` para especificar una cuenta de correo o un número de teléfono. Cuando se hace clic en un enlace de estas características, el sistema abre el programa encargado de responder a este tipo de solicitudes (enviar un correo o hacer una llamada telefónica) y le envía los datos especificados en el enlace. El siguiente ejemplo implementa la palabra clave `mailto` para enviar un correo electrónico.

---

```
<footer>
  <address>Barcelona, España</address>
  <small>&copy; 2023 <a href="mailto:hola@twooweb.com">Twooweb</a></small>
</footer>
```

---

**Listado 2-29:** *Enviando un correo electrónico*



**Ejercicio 2.8** :reemplace el elemento `<footer>` en su archivo `index.html` con el código del Listado 2-29. Abra el documento en su navegador y haga clic en el enlace. El sistema debería abrir su programa de correo para enviar un mensaje a la cuenta [hola@twooweb.com](mailto:hola@twooweb.com).

Los documentos a los que se accede a través de un enlace creado por el elemento `<a>` son descargados por el navegador y mostrados en pantalla, pero a veces los usuarios no necesitan que el navegador abra el documento, sino que el archivo se almacene en sus discos duros para usarlo más adelante. HTML ofrece dos atributos para este propósito:

**download**—Este es un atributo booleano que, cuando se incluye, indica que en lugar de leer el archivo el navegador debería descargarlo.

**ping**—Este atributo declara la ruta del archivo que se debe abrir en el servidor cuando el usuario hace clic en el enlace. El valor puede ser una o más URL separadas por un espacio.

Cuando el atributo `download` se encuentra presente dentro de un elemento `<a>`, el archivo especificado por el atributo `href` se descarga y almacena en el disco duro del usuario. Por otro lado, el archivo que indica el atributo `ping` no se descarga, sino que se ejecuta en el servidor. Este archivo se puede usar para ejecutar código que almacena información en el servidor cada vez que el archivo principal se descarga o para llevar un control de las veces que esta acción ocurre. En el siguiente ejemplo, implementamos ambos atributos para permitir al usuario descargar un archivo PDF.

---

```
<article>
  <p>La muerte es una quimera: porque mientras yo existo, no existe la
muerte;<br>y cuando existe la muerte, ya no existo yo.<br>Epicuro de
Samos</p>
  <footer>
    <a href="http://www.formasterminds.com/content/miarchivo.pdf"
ping="http://www.formasterminds.com/control.php" download>Clic aquí
para descargar</a>
  </footer>
</article>
```

---

**Listado 2-30:** Aplicando los atributos `ping` y `download`

En el ejemplo del Listado 2-30, agregamos un pie de página al artículo de ejemplos anteriores con un enlace a un archivo PDF. En circunstancias normales, un navegador moderno mostraría el contenido del archivo en pantalla, pero en este caso el atributo `download` obliga al navegador a descargar el archivo y almacenarlo en el disco duro.

Este ejemplo incluye un atributo `ping` que apunta a un archivo llamado `control.php`. Como resultado, cada vez que el usuario hace clic en el enlace, se descarga el archivo PDF y el código PHP se ejecuta en el servidor, lo que permite al desarrollador hacer un seguimiento de las veces que esta acción ocurre (almacenando información acerca del usuario en una base de datos, por ejemplo).



**IMPORTANTE:** el propósito del atributo `ping` es ejecutar código en el servidor. En el ejemplo hemos usado un archivo con código PHP, pero podría hacer lo mismo con cualquier otro lenguaje de programación que funcione en el servidor, como Python o Ruby. El modo de programar el archivo asignado al atributo `ping` depende del lenguaje que usemos y lo que queramos lograr. El tema va más allá del propósito de este libro. Para obtener más información sobre PHP, visite nuestro sitio web y siga los enlaces de este capítulo.

## Imágenes

Las imágenes pueden ser consideradas el segundo medio más importante en la Web. HTML incluye los siguientes elementos para introducir imágenes en nuestros documentos.

**<img>**—Este elemento inserta una imagen en el documento. El elemento requiere del atributo **src** para especificar la URL del archivo con la imagen que queremos incorporar.

**<picture>**—Este elemento inserta una imagen en el documento. Trabaja junto con el elemento **<source>** para ofrecer múltiples imágenes en diferentes resoluciones. Es útil para crear sitios web adaptables, como veremos en el Capítulo 5.

**<figure>**—Este elemento representa contenido asociado con el contenido principal, pero que se puede eliminar sin que se vea afectado, como fotos, vídeos, etc.

**<figcaption>**—Este elemento introduce un título para el elemento **<figure>**.

Para incluir una imagen en el documento, solo necesitamos declarar el elemento **<img>** y asignar la URL del archivo al atributo **src**.

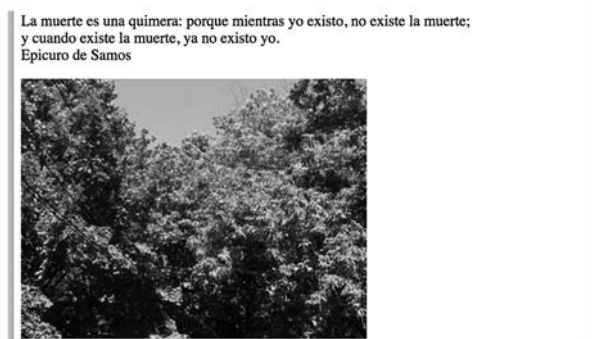
---

```
<article>
  <p>La muerte es una quimera: porque mientras yo existo, no existe la
muerte;<br>y cuando existe la muerte, ya no existo yo.<br>Epicuro de
Samos</p>
  
</article>
```

---

**Listado 2-31:** Incluyendo una imagen en el documento

El código del Listado 2-31 carga la imagen del archivo **miimagen.jpg** y la muestra en pantalla en su tamaño original, como lo ilustra la Figura 2-15.



**Figura 2-15:** Imagen en el documento

La imagen se representa en su tamaño original, pero podemos definir un tamaño personalizado y algunos parámetros de configuración usando el resto de los atributos disponibles para el elemento `<img>`.

**width**—Este atributo declara el ancho de la imagen.

**height**—Este atributo declara la altura de la imagen.

**alt**—Este atributo especifica el texto que se muestra cuando la imagen no se puede cargar.

**srcset**—Este atributo nos permite especificar una lista de imágenes de diferentes resoluciones que el navegador puede cargar para el mismo elemento.

**sizes**—Este atributo especifica una lista de *media queries* (consulta de medios) junto con distintos tamaños de imágenes para que el navegador decida qué mostrar según la resolución de la pantalla. Estudiaremos *media queries* y diseño web adaptable en el Capítulo 5.

**crossorigin**—Este atributo establece las credenciales para imágenes de origen cruzado (múltiples orígenes). Los valores posible son **anonymous** (sin credenciales) y **use-credentials** (requiere credenciales). Estudiaremos la tecnología CORS y cómo trabajar con imágenes procedentes de diferentes orígenes en el Capítulo 11.

Los atributos `width` y `height` determinan las dimensiones de la imagen, pero no tienen en cuenta la relación. Si declaramos ambos valores sin considerar la proporción original de la imagen, el navegador deberá estirar o achatar la imagen para adaptarla a las dimensiones definidas. Para reducir la imagen sin cambiar la proporción original, podemos especificar uno solo de los atributos y dejar que el navegador calcule el otro.

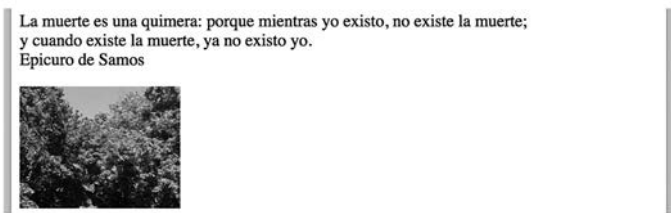
---

```
<article>
  <p>La muerte es una quimera: porque mientras yo existo, no existe la
muerte;<br>y cuando existe la muerte, ya no existo yo.<br>Epicuro de
Samos</p>
  
</article>
```

---

**Listado 2-32:** Reduciendo el tamaño de la imagen

El código en el Listado 2-32 agrega el atributo `width` con el valor 150 al elemento `<img>` introducido en el ejemplo anterior. Esto reduce el ancho de la imagen a 150 píxeles, pero como el atributo `height` no se ha declarado, la altura de la imagen se calcula automáticamente considerando las proporciones originales de la imagen. El resultado se muestra en la Figura 2-16.



**Figura 2-16:** Imagen con un tamaño personalizado

**Ejercicio 2.9 :**descargue una imagen y renómbrela como miimagen.jpg.

Reemplace el elemento `<article>` en su archivo `index.html` por el código del Listado 2-31 y abra el documento en su navegador. Debería ver algo similar a la Figura 2-15. Repita el proceso con el código del Listado 2-32. Ahora debería ver la imagen reducida, según se ilustra en la Figura 2-16. El elemento `<img>` en el código del Listado 2-32 también incluye el atributo `alt`. Para probar este atributo, borre el archivo `miimagen.jpg` y actualice el documento en su navegador. Como el navegador ya no puede encontrar el archivo de la imagen, mostrará el texto asignado al atributo `alt` en su lugar.

Algunas imágenes, como los iconos, son importantes porque otorgan un significado al resto del contenido, pero otras, como la imagen de estos ejemplos, actúan como complemento y se pueden eliminar sin que afecten al flujo de información. Cuando esta clase de información se encuentra presente, se puede utilizar el elemento `<figure>` para identificarla. Este elemento se suele implementar junto con el elemento `<figcaption>` para incluir texto descriptivo. En el siguiente ejemplo usamos estos dos elementos para identificar nuestra imagen y mostrar su título al usuario.

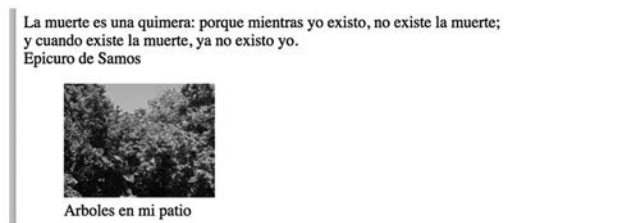
---

```
<article>
  <p>La muerte es una quimera: porque mientras yo existo, no existe la
muerte;<br>y cuando existe la muerte, ya no existo yo.<br>Epicuro de
Samos</p>
  <figure>
    
    <figcaption>Arboles en mi patio</figcaption>
  </figure>
</article>
```

---

**Listado 2-33:** Identificando la imagen con el elemento `<figure>`

Por defecto, los navegadores asignan márgenes laterales al elemento `<figure>`. El resultado se muestra en la Figura 2-17.



**Figura 2-17:** Imagen junto a su título

## Listados

A menudo, la información se debe representar como una lista de ítems. Por ejemplo, muchos sitios web incluyen listados de libros, películas, o términos y descripciones. Para crear estos listados, HTML ofrece los siguientes elementos.

**<ul>**—Este elemento crea una lista de ítems sin orden. Está compuesto por etiquetas de apertura y cierre para agrupar los ítems (**<ul>** y **</ul>**) y trabaja junto con el elemento **<li>** para definir cada uno de los ítems de la lista.

**<ol>**—Este elemento crea una lista ordenada de ítems. Está compuesto por etiquetas de apertura y cierre para agrupar los ítems (**<ol>** y **</ol>**) y trabaja junto con el elemento **<li>** para definir los ítems de la lista. Este elemento puede incluir los atributos **reversed** para invertir el orden de los indicadores, **start** para determinar el valor desde el cual los indicadores tienen que comenzar a contar y **type** para determinar el tipo de indicador que queremos usar. Los valores disponibles para el atributo **type** son **1** (números), **a** (letras minúsculas), **A** (letras mayúsculas), **i** (números romanos en minúsculas) e **I** (números romanos en mayúsculas).

**<dl>**—Este elemento crea una lista de términos y descripciones. El elemento trabaja junto con los elementos **<dt>** y **<dd>** para definir los ítems de la lista. El elemento **<dl>** define la lista, el elemento **<dt>** define los términos y el elemento **<dd>** define las descripciones.

El elemento que usamos para crear la lista depende de las características del contenido. Por ejemplo, si el orden de los ítems no es importante, podemos usar el elemento **<ul>**. En esta clase de listas, los ítems se declaran entre las etiquetas **<ul>** con el elemento **<li>**, como muestra el siguiente ejemplo.

```
<aside>
  <ul>
    <li>IT, Stephen King</li>
    <li>Carrie, Stephen King</li>
    <li>El Resplandor, Stephen King</li>
    <li>Misery, Stephen King</li>
  </ul>
</aside>
```

**Listado 2-34:** Creando una lista de ítems sin orden

El elemento **<ul>** presenta los ítems en el orden en el que se han declarado en el código y los identifica con un punto del lado izquierdo.

- IT, Stephen King
- Carrie, Stephen King
- El Resplandor, Stephen King
- Misery, Stephen King

**Figura 2-18:** Lista sin orden

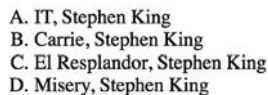
Si necesitamos declarar la posición de cada ítem, podemos crear la lista con el elemento `<ol>`. Este elemento crea una lista de ítems en el orden en el que se han declarado en el código, pero en lugar de usar puntos para identificarlos, les asigna un valor. Por defecto, los indicadores se crean con números, pero podemos cambiarlos con el atributo `type`. En el siguiente ejemplo, usamos letras mayúsculas.

---

```
<aside>
  <ol type="A">
    <li>IT, Stephen King</li>
    <li>Carrie, Stephen King</li>
    <li>El Resplandor, Stephen King</li>
    <li>Misery, Stephen King</li>
  </ol>
</aside>
```

---

**Listado 2-35:** Creando una lista ordenada de ítems



A. IT, Stephen King  
B. Carrie, Stephen King  
C. El Resplandor, Stephen King  
D. Misery, Stephen King

**Figura 2-19:** Lista ordenada

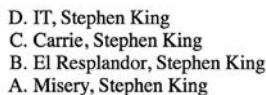
Aunque los ítems se muestran siempre en el orden en el que se han declarado en el código, podemos utilizar otros atributos del elemento `<ol>` para cambiar el orden de los indicadores. Por ejemplo, agregando el atributo **reversed** logramos que los indicadores se muestren en orden invertido.

---

```
<aside>
  <ol type="A" reversed>
    <li>IT, Stephen King</li>
    <li>Carrie, Stephen King</li>
    <li>El Resplandor, Stephen King</li>
    <li>Misery, Stephen King</li>
  </ol>
</aside>
```

---

**Listado 2-36:** Creando una lista en orden invertido



D. IT, Stephen King  
C. Carrie, Stephen King  
B. El Resplandor, Stephen King  
A. Misery, Stephen King

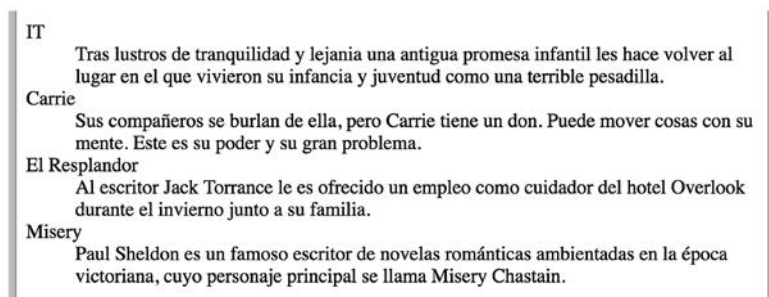
**Figura 2-20:** Lista en orden invertido

Los elementos <dl>, <dt>, y <dd> trabajan de forma similar al elemento <ul>, pero su propósito es mostrar una lista de términos y descripciones. Los términos se representan por el elemento <dt> y las descripciones por el elemento <dd>. En el siguiente ejemplo, los usamos para agregar la descripción de los libros listados anteriormente.

```
<aside>
  <dl>
    <dt>IT</dt>
    <dd>Tras lustros de tranquilidad y lejanía una antigua promesa infantil les hace volver al lugar en el que vivieron su infancia y juventud como una terrible pesadilla.</dd>
    <dt>Carrie</dt>
    <dd>Sus compañeros se burlan de ella, pero Carrie tiene un don. Puede mover cosas con su mente. Este es su poder y su gran problema.</dd>
    <dt>El Resplandor</dt>
    <dd>Al escritor Jack Torrance le es ofrecido un empleo como cuidador del hotel Overlook durante el invierno junto a su familia.</dd>
    <dt>Misery</dt>
    <dd>Paul Sheldon es un famoso escritor de novelas románticas ambientadas en la época victoriana, cuyo personaje principal se llama Misery Chastain.</dd>
  </dl>
</aside>
```

**Listado 2-37:** Creando una lista de términos y descripciones

Por defecto, los navegadores otorgan estilos a este tipo de listas que incluyen márgenes a los lados para diferenciar los términos de las descripciones. El resultado se muestra en la Figura 2-21.



**Figura 2-21:** Lista de términos y descripciones



**Ejercicio 2.10:** reemplace el elemento <aside> en su archivo index.html con el ejemplo que quiere probar y abra el documento en su navegador. Inserte el atributo **start** con el valor "3" en el elemento <ol> del Listado 2-35. Los indicadores deberían comenzar a contar desde la letra C.



Los siguientes elementos se han diseñado con propósitos diferentes, pero también se utilizan frecuentemente para construir listas de ítems.

**<blockquote>**—Este elemento representa un bloque de texto que incluye una cita tomada de otro texto en el documento.

**<details>**—Este elemento crea una herramienta que se expande cuando se hace clic en ella para mostrar información adicional. La parte visible se define con el elemento

**<summary>**, y se pueden usar elementos comunes como **<p>** para definir el contenido.

El elemento **<blockquote>** es similar al elemento **<p>**, pero como también incluye márgenes a los costados, se ha usado tradicionalmente para presentar listas de valores, como lo demuestra el siguiente ejemplo.

---

```
<aside>
  <blockquote>IT</blockquote>
  <blockquote>Carrie</blockquote>
  <blockquote>El Resplandor</blockquote>
  <blockquote>Misery</blockquote>
</aside>
```

---

**Listado 2-38:** Creando una lista con el elemento **<blockquote>**

El listado generado por el elemento **<blockquote>** se muestra igual que otras listas, pero no incluye puntos o números para identificar cada ítem.

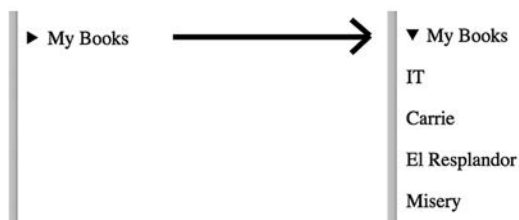


**Figura 2-22:** Listado de ítems con el elemento **<blockquote>**

En sitios web modernos son comunes las herramientas que revelan información adicional cuando el usuario lo requiere. Para ofrecer esta posibilidad, HTML incluye el elemento **<details>**. Este elemento muestra un título, especificado por el elemento **<summary>**, y contenido que se puede representar por elementos comunes como **<p>** o **<blockquote>**. Debido a esto, el elemento **<details>** se puede utilizar para revelar una lista de valores, como lo hacemos en el siguiente ejemplo.

```
<details>
  <summary>My Books</summary>
  <p>IT</p>
  <p>Carrie</p>
  <p>El Resplandor</p>
  <p>Misery</p>
</details>
```

**Listado 2-39:** Revelando información con los elementos `<details>` y `<summary>`



**Figura 2-23:** El elemento `<details>` antes y después de ser clicado

## Tablas

Las tablas organizan información en filas y columnas. Debido a sus características, se usaron durante mucho tiempo para estructurar documentos HTML, pero con la introducción de CSS, los desarrolladores pudieron lograr el mismo efecto implementando otros elementos. Aunque ya no se recomienda usar tablas para definir la estructura de un documento, todavía se utilizan para presentar información tabular, como estadísticas o especificaciones técnicas, por ejemplo. HTML incluye varios elementos para crear una tabla. Los siguientes son los más utilizados.

**<table>**—Este elemento define una tabla. Incluye etiquetas de apertura y cierre para agrupar el resto de los elementos que definen la tabla.

**<tr>**—Este elemento define una fila de celdas. Incluye etiquetas de apertura y cierre para agrupar las celdas.

**<td>**—Este elemento define una celda. Incluye etiquetas de apertura y cierre para delimitar el contenido de la celda y puede incluir los atributos `colspan` y `rowspan` para indicar cuántas columnas y filas ocupa la celda.

**<th>**—Este elemento define una celda para la cabecera de la tabla. Incluye etiquetas de apertura y cierre para delimitar el contenido de la celda y puede incluir los atributos `colspan` y `rowspan` para indicar cuántas columnas y filas ocupa la celda.

Para incluir una tabla en el documento, primero tenemos que declarar el elemento `<table>` y luego describir las filas una por una con los elementos `<tr>` y `<td>`, como muestra el siguiente ejemplo.

```
<article>
  <table>
    <tr>
      <td>IT</td>
      <td>Stephen King</td>
      <td>1986</td>
    </tr>
    <tr>
      <td>Carrie</td>
      <td>Stephen King</td>
      <td>1974</td>
    </tr>
    <tr>
      <td>El Resplandor</td>
      <td>Stephen King</td>
      <td>1977</td>
    </tr>
  </table>
</article>
```

**Listado 2-40:** *Creando una tabla*

Debido a que el navegador interpreta el documento de forma secuencial desde la parte superior a la inferior, cada vez que declaramos una fila, tenemos que declarar las celdas que corresponden a esa fila y su contenido. Siguiendo este patrón, en el Listado 2-40, creamos una tabla para mostrar libros, uno por fila. La primer celda de cada fila representa el título del libro, la segunda celda representa el autor, y la tercera celda el año de publicación. Cuando el navegador abre este documento, muestra la información en el orden en el que se ha declarado en el código y con el tamaño determinado por el contenido de las celdas.

IT	Stephen King	1986
Carrie	Stephen King	1974
El Resplandor	Stephen King	1977

**Figura 2-24:** *Tabla con estilos por defecto*

Si queremos incluir una cabecera para describir el contenido de cada columna, podemos crear una fila de celdas adicional representadas con el elemento <th>.

```
<article>
<table>
  <tr>
    <th>Título</th>
    <th>Autor</th>
    <th>Año</th>
  </tr>
  <tr>
    <td>IT</td>
    <td>Stephen King</td>
    <td>1986</td>
  </tr>
  <tr>
    <td>Carrie</td>
    <td>Stephen King</td>
    <td>1974</td>
  </tr>
  <tr>
    <td>El Resplandor</td>
    <td>Stephen King</td>
    <td>1977</td>
  </tr>
</table>
</article>
```

**Listado 2-41:** Creando una tabla con cabecera

Por defecto, los navegadores muestran las cabeceras con el texto en negrita y centrado.



Título	Autor	Año
IT	Stephen King	1986
Carrie	Stephen King	1974
El Resplandor	Stephen King	1977

**Figura 2-25:** Cabecera de tabla con estilos por defecto

Las celdas se pueden estirar para que ocupen más de una columna con los atributos colspan y rowspan. Por ejemplo, podemos usar solo una celda de cabecera para identificar el título y el autor del libro.

---

```

<article>
  <table>
    <tr>
      <th colspan="2">Libro</th>
      <th>Año</th>
    </tr>
    <tr>
      <td>IT</td>
      <td>Stephen King</td>
      <td>1986</td>
    </tr>
    <tr>
      <td>Carrie</td>
      <td>Stephen King</td>
      <td>1974</td>
    </tr>
    <tr>
      <td>El Resplandor</td>
      <td>Stephen King</td>
      <td>1977</td>
    </tr>
  </table>
</article>

```

---

**Listado 2-42:** Estirando celdas

El ejemplo del Listado 2-42 incluye una celda de cabecera con el título Libro para las primeras dos columnas. Debido al valor asignado al atributo colspan, esta celda se estira para que ocupe el espacio de dos. El resultado se muestra en la Figura 2-26.

Libro		Año
IT	Stephen King	1986
Carrie	Stephen King	1974
El Resplandor	Stephen King	1977

**Figura 2-26:** Celdas de múltiples columnas



**Lo básico:** HTML también incluye los elementos `<thead>`, `<tbody>`, y `<tfoot>` para representar la cabecera, el cuerpo, y el pie de la tabla, respectivamente, y otros elementos como `<colgroup>` para agrupar columnas. Para obtener más información, visite nuestro sitio web y siga los enlaces de este capítulo.

## Atributos Globales

La mayoría de los navegadores actuales automáticamente traducen el contenido del documento cuando detectan que se ha escrito en un idioma diferente al del usuario, pero en algunos casos la página puede incluir frases o párrafos enteros que no se deben alterar, como nombres de personas o títulos de películas. Para controlar el proceso de traducción, HTML ofrece un atributo global llamado `translate`. Este atributo puede tomar dos valores: `yes` y `no`. Por defecto, el valor es `yes` (sí). En el siguiente ejemplo, usamos un elemento `<span>` para especificar la parte del texto que no se debería traducir.

---

```
<p>My favorite movie is <span translate="no">Two Roads</span></p>
```

---

**Listado 2-43:** Usando el atributo `translate`

El elemento `<span>` se diseñó para presentar texto, pero a diferencia del elemento `<p>` no asigna ningún estilo al contenido, por lo que el texto se presenta en la misma línea y con el tipo de letra y tamaño por defecto. La Figura 2-27 muestra lo que vemos en la ventana del navegador después de traducirlo.



Mi película favorita es Two Roads

**Figura 2-27:** texto traducido por el navegador



**Ejercicio 2.11** :inserte el código del Listado 2-43 en la sección principal de su documento y abra el documento en su navegador. Los navegadores como Google Chrome ofrecen una opción para traducir el documento en un menú contextual cuando hacemos clic con el botón derecho del ratón. Si la traducción no se realiza de forma automática, seleccione esta opción para traducir el texto. Una vez traducido el texto, debería ver algo similar a lo que se muestra en la Figura 2-27.

Otro atributo útil que podemos agregar a un elemento HTML es `contenteditable`. Este es un atributo booleano que, si está presente, permite al usuario editar el contenido del elemento. Si el usuario hace clic en un elemento que contiene este atributo, puede cambiar su contenido. El siguiente ejemplo implementa el elemento `<span>` nuevamente para permitir a los usuarios editar el nombre de una película.

---

```
<p>Mi película favorita es <span contenteditable>Casablanca</span></p>
```

---

**Listado 2-44:** Usando el atributo `contenteditable` para editar contenido



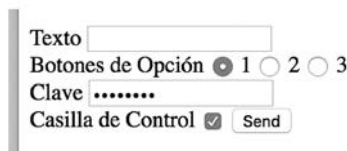
**Ejercicio 2.12 :**reemplace el párrafo del Listado 2-43 por el párrafo del Listado 2-44 y abra el documento en su navegador. Haga clic en el nombre de la película para cambiarlo.



**IMPORTANTE:** las modificaciones introducidas por el usuario solo se encuentran disponibles en su ordenador. Si queremos que los cambios se almacenen en el servidor, tenemos que subir esta información con un programa en JavaScript usando Ajax, según veremos en el Capítulo 21.

### 3. Formularios

Los formularios son herramientas que podemos incluir en un documento para permitir a los usuarios insertar información, tomar decisiones, comunicar datos y cambiar el comportamiento de una aplicación. El propósito principal de los formularios es permitir al usuario seleccionar o insertar información y enviarla al servidor para ser procesada. La Figura 2-28 muestra algunas de las herramientas facilitadas este fin.

A screenshot of a web form. It contains a text input field labeled 'Texto', a group of radio buttons labeled 'Botones de Opción' with options 1, 2, and 3 (option 1 is selected), a password input field labeled 'Clave' with masked characters, a checkbox labeled 'Casilla de Control' which is checked, and a 'Send' button.

**Figura 2-28:** Formulario en el navegador

#### Definición

Como se muestra en la Figura 2-28, los formularios pueden presentar varias herramientas que permiten al usuario interactuar con el documento, incluidos campos de texto, casillas de control, menús desplegables y botones. Cada una de estas herramientas se representa por un elemento y el formulario queda definido por el elemento `<form>`, que incluye etiquetas de apertura y cierre para agrupar al resto de los elementos y requiere de algunos atributos para determinar cómo se envía la información al servidor.

**name**—Este atributo especifica el nombre del formulario. También se encuentra disponible para otros elementos, pero es particularmente útil para elementos de formulario, como veremos más adelante.

**method**—Este atributo determina el método a utilizar para enviar la información al servidor. Existen dos valores disponibles: `GET` y `POST`. El método `GET` se usa para enviar una cantidad limitada de información de forma pública (los datos son incluidos en la URL, la cual no puede contener más de 255 caracteres). Por otro lado, el método `POST` se utiliza para enviar una cantidad ilimitada de información de forma privada (los datos no son visibles al usuario y pueden tener la longitud que necesitemos).

**action**—Este atributo declara la URL del archivo en el servidor que va a procesar la información enviada por el formulario.

**target**—Este atributo determina dónde se mostrará la respuesta recibida desde el servidor. Los valores disponibles son `_blank` (nueva ventana), `_self` (mismo recuadro), `_parent` (recuadro padre), y `_top` (la ventana que contiene el recuadro). El valor `_self` se declara por defecto, lo que significa que la respuesta recibida desde el servidor se mostrará en la misma ventana.

**enctype**—Este atributo declara la codificación aplicada a los datos que envía el formulario. Puede tomar tres valores: `application/x-www-form-urlencoded` (los caracteres son codificados), `multipart/form-data` (los caracteres no son codificados), `text/plain` (solo los espacios son codificados). El primer valor se asigna por defecto.



**accept-charset**—Este atributo declara el tipo de codificación aplicada al texto del formulario. Los valores más comunes son **UTF-8** e **ISO-8859-1**. El valor por defecto se asigna al documento con el elemento **<meta>** (ver Listado 2-6).

El siguiente ejemplo define un formulario básico. El atributo **name** identifica el formulario con el nombre "formulario", el atributo **method** determina que los datos se incluirán en la URL (GET), y el atributo **action** declara que **procesar.php** es el archivo que se ejecutará en el servidor para procesar la información y devolver el resultado.

---

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="utf-8">
  <title>Formularios</title>
</head>
<body>
  <section>
    <form name="formulario" method="get" action="procesar.php">

    </form>
  </section>
</body>
</html>
```

---

**Listado 2-45:** Definiendo un formulario con el elemento `<form>`



**Ejercicio 2.13** : cree un nuevo archivo HTML en su editor o modifique el archivo del ejemplo anterior con el código del Listado 2-45 y abra el documento en su navegador. En este momento no verá nada en la pantalla porque el formulario se ha declarado vacío. A continuación, desarrollaremos su contenido.

## Elementos

Un formulario puede incluir diferentes herramientas para permitir al usuario seleccionar o insertar información. HTML incluye múltiples elementos para crear estas herramientas. Los siguientes son los más utilizados.

**<input>**—Este elemento crea un campo de entrada. Puede recibir diferentes tipos de entradas, dependiendo del valor del atributo **type**.

**<textarea>**—Este elemento crea un campo de entrada para insertar múltiples líneas de texto. El tamaño se puede declarar en números enteros usando los atributos **rows** y **cols**, o en píxeles con estilos CSS, como veremos en el Capítulo 3.

**<select>**—Este elemento crea una lista de opciones que el usuario puede elegir. Trabaja junto con el elemento **<option>** para definir cada opción y el elemento **<optgroup>** para organizar las opciones en grupos.

**<button>**—Este elemento crea un botón. Incluye el atributo **type** para definir el propósito del botón. Los valores disponibles son **submit** para enviar el formulario (por defecto), **reset** para reiniciar el formulario, y **button** para realizar tareas personalizadas.

**<output>**—Este elemento representa un resultado producido por el formulario. Se implementa por medio de código JavaScript para mostrar el resultado de una operación al usuario.

**<meter>**—Este elemento representa una medida o el valor actual de un rango.

**<progress>**—Este elemento representa el progreso de una operación.

**<datalist>**—Este elemento crea un listado de valores disponibles para otros controles. Trabaja junto con el elemento **<option>** para definir cada valor.

**<label>**—Este elemento crea una etiqueta para identificar un elemento de formulario.

**<fieldset>**—Este elemento agrupa otros elementos de formulario. Se usa para crear secciones dentro de formularios extensos. El elemento puede contener un elemento **<legend>** para definir el título de la sección.

El elemento **<input>** es el más versátil de todos. Este elemento genera un campo de entrada en el que el usuario puede seleccionar o insertar información, pero puede adoptar diferentes características y aceptar varios tipos de valores dependiendo del valor de su atributo **type**. Los siguientes son los valores disponibles para este atributo.

**text**—Este valor genera un campo de entrada para insertar texto genérico. **email**—

Este valor genera un campo de entrada para insertar cuentas de correo. **search**—

Este valor genera un campo de entrada para insertar términos de búsqueda. **url**—Este valor genera un campo de entrada para insertar URL.

**tel**—Este valor genera un campo de entrada para insertar números de teléfono.

**number**—Este valor genera un campo de entrada para insertar números. **range**—

Este valor genera un campo de entrada para insertar un rango de números. **date**—

Este valor genera un campo de entrada para insertar una fecha.

**datetime-local**—Este valor genera un campo de entrada para insertar fecha y hora. **week**—Este valor genera un campo de entrada para insertar el número de la semana (dentro del año).

**month**—Este valor genera un campo de entrada para insertar el número del mes.

**time**—Este valor genera un campo de entrada para insertar una hora (horas y minutos).

**hidden**—Este valor oculta el campo de entrada. Se usa para enviar información complementaria al servidor.

**password**—Este valor genera un campo de entrada para insertar una clave. Reemplaza los caracteres insertados con estrellas o puntos para ocultar información sensible.

**color**—Este valor genera un campo de entrada para insertar un color.

**checkbox**—Este valor genera una casilla de control que permite al usuario activar o desactivar una opción.

**radio**—Este valor genera un botón de opción para seleccionar una opción de varias posibles.

**file**—Este valor genera un campo de entrada para seleccionar un archivo en el ordenador del usuario.

**button**—Este valor genera un botón. El botón trabaja como el elemento `<button>` de tipo `button`. No realiza ninguna acción por defecto; la acción debe ser definida desde JavaScript, como veremos en próximos capítulos.

**submit**—Este valor genera un botón para enviar el formulario.

**reset**—Este valor genera un botón para reiniciar el formulario.

**image**—Este valor carga una imagen que se usa como botón para enviar el formulario. Un elemento `<input>` de este tipo debe incluir el atributo `src` para especificar la URL de la imagen.

Para incluir un formulario en nuestro documento, tenemos que declararlo con el elemento `<form>`, como hemos hecho en el ejemplo anterior, y luego incorporar en su interior todos los elementos que el usuario necesitará para insertar la información y enviarla al servidor. Por ejemplo, si queremos que el usuario inserte su nombre y edad, tenemos que incluir dos campos de entrada para texto y un tercer elemento para crear el botón con el que enviar el formulario.

---

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="utf-8">
  <title>Formularios</title>
</head>
<body>
  <section>
    <form name="formulario" method="get" action="procesar.php">
      <p><input type="text" name="nombre"></p>
      <p><input type="text" name="edad"></p>
      <p><input type="submit"></p>
    </form>
  </section>
</body>
</html>
```

---

**Listado 2-46:** Incluyendo herramientas en un formulario

La información insertada en el formulario se envía al servidor para ser procesada. Para que el servidor pueda identificar cada valor, los elementos deben incluir el atributo name. Con este atributo podemos asignar un nombre único a cada elemento. En el ejemplo del Listado 2-46, llamamos a los campos de entrada "nombre" y "edad" (el elemento <input> que crea el botón para enviar el formulario no necesita un nombre porque no envía ningún dato al servidor).

Los elementos de formulario no generan un salto de línea; se muestran en la pantalla uno detrás del otro. Si queremos que el navegador muestre un elemento en cada línea, tenemos que modificar el diseño nosotros mismos. En el Listado 2-46, usamos elementos <p> para separar los elementos del formulario, pero este diseño normalmente se logra a través de estilos CSS, como veremos en el Capítulo 4. El resultado se muestra en la Figura 2-29.

The image shows a simple web form layout. It consists of two rectangular text input fields stacked vertically. Below the second input field is a button with the text "Submit". The entire form is enclosed within a thin rectangular border.

**Figura 2-29:** Formulario con dos campos de entrada y un botón para enviar los datos

Otro atributo que podemos usar en este ejemplo es value. El tipo de entrada submit crea un botón para enviar el formulario. Por defecto, los navegadores le dan al botón el título Enviar (Submit), pero podemos usar el atributo value para modificarlo.

---

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="utf-8">
  <title>Formularios</title>
</head>
<body>
  <section>
    <form name="formulario" method="get" action="procesar.php">
      <p><input type="text" name="nombre"></p>
      <p><input type="text" name="edad"></p>
      <p><input type="submit" value="Enviar Datos"></p>
    </form>
  </section>
</body>
</html>
```

---

**Listado 2-47:** Asignando un título diferente para el botón Enviar

El atributo value también se puede usar para declarar el valor inicial de un elemento. Por ejemplo, podemos insertar la edad del usuario en el campo edad si ya la conocemos.

---

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="utf-8">
  <title>Formularios</title>
</head>
<body>
  <section>
    <form name="formulario" method="get" action="procesar.php">
      <p><input type="text" name="nombre"></p>
      <p><input type="text" name="edad" value="35"></p>
      <p><input type="submit" value="Enviar Datos"></p>
    </form>
  </section>
</body>
</html>
```

---

**Listado 2-48:** Declarando valores iniciales



**Ejercicio 2.14 :** copie el código del Listado 2-48 dentro de su archivo HTML y abra el documento en su navegador. Debería ver un formulario similar al de la Figura 2-29 pero con el número 35 dentro del campo **edad** y el botón para enviar el formulario con el título **Enviar datos**.

Los formularios necesitan incluir descripciones que le indiquen al usuario los datos que debe introducir. Por esta razón, HTML incluye el elemento `<label>`. Debido a que estos elementos no solo le indican al usuario qué valor debe introducir, sino que además ayudan al navegador a identificar cada parte del formulario, tienen asociarse al elemento al que están describiendo. Para asociar un elemento `<label>` con el elemento de formulario correspondiente, podemos incluir el elemento de formulario dentro del elemento `<label>`, como muestra el siguiente ejemplo.

---

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="utf-8">
  <title>Formularios</title>
</head>
<body>
  <section>
    <form name="formulario" method="get" action="procesar.php">
      <p>
        <label>Nombre: <input type="text" name="nombre"></label>
      </p>
      <p>
        <label>Edad: <input type="text" name="edad"></label>
      </p>
      <p><input type="submit" value="Enviar"></p>
    </form>
  </section>
</body>
</html>
```

---

***Listado 2-49: Identificando elementos de formulario***

Otra alternativa para asociar un elemento `<label>` con su elemento de formulario es implementando el atributo `for`. El atributo `for` conecta el elemento `<label>` con el elemento de formulario por medio del valor del atributo `id`, según ilustra el siguiente ejemplo.

---

```

<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="utf-8">
    <title>Formularios</title>
</head>
<body>
    <section>
        <form name="formulario" method="get" action="procesar.php">
            <p>
                <label for="nombre">Nombre: </label>
                <input type="text" name="nombre" id="nombre"></input>
            </p>
            <p>
                <label for="edad">Edad: </label>
                <input type="text" name="edad" id="edad"></input>
            </p>
            <p><input type="submit" value="Enviar"></p>
        </form>
    </section>
</body>
</html>

```

---

**Listado 2-50:** Asociando etiquetas con elementos por medio del atributo `for`

El elemento `<label>` no incluye ningún estilo por defecto; lo único que hace es asociar una etiqueta con un elemento y, por lo tanto, el texto se muestra en pantalla con el tipo de letra y el tamaño por defecto.



The image shows a simple web form rendered in a browser. It contains two text input fields. The first field is preceded by the label 'Nombre:' and the second by 'Edad:'. Below these fields is a submit button with the text 'Enviar'. The labels are in a standard sans-serif font, and the input fields are simple rectangular boxes. The overall layout is clean and functional.

**Figura 2-30:** Elementos identificados con una etiqueta

Los campos de entrada usados en los ejemplos anteriores eran de tipo `text`, lo que significa que los usuarios pueden introducir cualquier clase de texto que deseen, pero esto no es lo que necesitamos para este formulario. El primer campo espera un nombre, por lo que no debería permitir que se introduzcan números o textos muy extensos, y el segundo campo espera la edad del usuario, por lo que no debería aceptar ningún tipo de carácter excepto números. Para determinar cuántos caracteres se pueden introducir, el elemento `<input>` debe incluir los siguientes atributos.

**maxlength**—Este atributo especifica el máximo número de caracteres que se permite introducir en el campo.

**minlength**—Este atributo especifica el mínimo número de caracteres que se permite introducir en el campo.

El siguiente ejemplo limita el nombre a un máximo de 15 caracteres.

---

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="utf-8">
  <title>Formularios</title>
</head>
<body>
  <section>
    <form name="formulario" method="get" action="procesar.php">
      <p><label>Nombre: <input type="text" name="nombre"
maxlength="15"></label></p>
      <p><label>Edad: <input type="text" name="edad"></label></p>
      <p><input type="submit" value="Enviar"></p>
    </form>
  </section>
</body>
</html>
```

---

**Listado 2-51:** Declarando el máximo número de caracteres permitidos

El atributo `maxlength` implementado en el formulario del Listado 2-51 limita el número de caracteres que el usuario puede introducir, pero el tipo de campo es aún `text`, lo que significa que en el campo se puede escribir cualquier valor. Si el usuario escribe un número en el campo nombre o letras en el campo edad, el navegador considerará la entrada válida. Para controlar lo que el usuario puede introducir, tenemos que declarar un tipo de campo diferente con el atributo `type`. El siguiente ejemplo declara el tipo `number` para el campo edad para permitir que solo se introduzcan números, e incluye otros campos para que el usuario pueda declarar su cuenta de correo, número de teléfono y sitio web.

---

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="utf-8">
  <title>Formularios</title>
</head>
<body>
  <section>
    <form name="formulario" method="get" action="procesar.php">
      <p><label>Nombre: <input type="text" name="nombre"
maxlength="15"></label></p>
      <p><label>Edad: <input type="number" name="edad"></label></p>
      <p><label>Correo: <input type="email" name="correo"></label></p>
      <p><label>Teléfono: <input type="tel" name="telefono"></label></p>
      <p><label>Sitio Web: <input type="url" name="sitioweb"></label></p>
      <p><input type="submit" value="Enviar"></p>
    </form>
  </section>
</body>
</html>
```

---

**Listado 2-52:** Solicitando tipos de entrada específicos

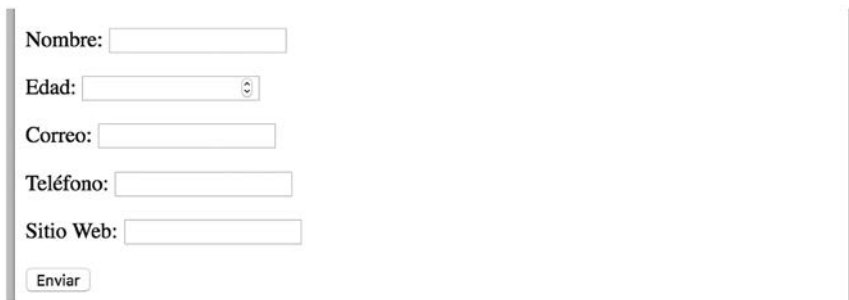


El tipo number asignado al campo edad en el Listado 2-52 le dice al elemento que solo acepte números. El resto de los tipos de entrada implementados en este ejemplo no imponen ninguna restricción en los caracteres introducidos, pero le indican al navegador la clase de valores que se esperan del usuario.

Por ejemplo, el tipo email espera una cuenta de correo, de modo que si el dato introducido no es una cuenta de correo, el navegador no permite que se envíe el formulario y muestra un error en pantalla. El tipo url trabaja exactamente igual que el tipo email, pero con direcciones web. Este tipo de campo acepta solo URL absolutas y devuelve un error si el valor no es válido.

Otros tipos como tel no exigen ninguna sintaxis en particular pero le solicitan al navegador que sugiera al usuario posibles valores a introducir o incluya un teclado específico en los dispositivos que lo requieren (en dispositivos móviles, el teclado que se muestra cuando el usuario hace clic en el campo telefono incluye solo dígitos para facilitar que se introduzcan números telefónicos).

Aunque estos tipos de campo presentan sus propias restricciones, todos se ven iguales en el navegador.

El formulario muestra cinco campos de entrada con etiquetas: 'Nombre:', 'Edad:', 'Correo:', 'Teléfono:' y 'Sitio Web:'. El campo 'Edad:' tiene un icono de flecha hacia abajo a su derecha. Debajo de los campos hay un botón etiquetado 'Enviar'.

**Figura 2-31:** Campos de entrada de diferentes tipos

Como ilustra la Figura 2-31, por defecto los navegadores incluyen flechas en el lado derecho de un campo de tipo number con las que podemos seleccionar un número. Para establecer restricciones en los números que el usuario puede seleccionar con estas flechas o controlar los números permitidos, los elementos `<input>` de este tipo pueden incluir los siguientes atributos.

**min**—El valor de este atributo determina el valor mínimo que acepta el campo.

**max**—El valor de este atributo determina el valor máximo que acepta el campo.

**step**—El valor de este atributo determina el número por el cual el valor del campo se puede incrementar o reducir. Por ejemplo, si declaramos el valor 5 para este atributo y un valor mínimo de 0 y uno máximo de 10 para el campo, el navegador no nos dejará introducir valores entre 0 y 5 o 5 y 10.

El siguiente ejemplo restringe el valor insertado en el campo edad de nuestro formulario a un mínimo de 13 y un máximo de 100.

---

```
<input type="number" name="edad" min="13" max="100">
```

---

**Listado 2-53:** *Restringiendo los números*

Otro tipo de entrada que implementa estos mismo atributos es range. El tipo range crea un campo que nos permite seleccionar un número desde un rango de valores. El valor inicial se establece de acuerdo a los valores de los atributos min y max, pero podemos declarar un valor específico con el atributo value, como muestra el siguiente ejemplo.

---

```
<input type="range" name="edad" min="13" max="100" value="35">
```

---

**Listado 2-54:** *Implementando el tipo range*

Los navegadores muestran un campo de entrada de tipo range como un control que el usuario puede deslizar para seleccionar un valor.



**Figura 2-32:** *Campo de entrada de tipo range*



**Ejercicio 2.15 :**reemplace el elemento `<input>` en su archivo HTML con el elemento de los Listados 2-53 o 2-54 para probar estos ejemplos. Abra el documento en su navegador e inserte o seleccione un número para ver cómo trabajan estos tipos de entrada. Intente enviar el formulario con un valor menor o mayor a los permitidos. El navegador debería mostrarle un error.

Los valores para el tipo range implementado en el ejemplo anterior no se introducen en un campo de texto, sino que se seleccionan desde una herramienta visual generada por el navegador. El tipo range no es el único que presenta esta clase de herramientas. Por ejemplo, el tipo radio crea un botón circular que se resalta cuando se selecciona (ver Figura 2-28). Esto nos permite crear una lista de valores que el usuario puede seleccionar con solo hacer clic en el botón correspondiente. Para ofrecer al usuario todas las opciones disponibles, tenemos que insertar un elemento `<input>` por cada opción. Los elementos `<input>` se asocian entre ellos por medio del valor del atributo name, y el valor de cada opción se define por el atributo value, como muestra el siguiente ejemplo.

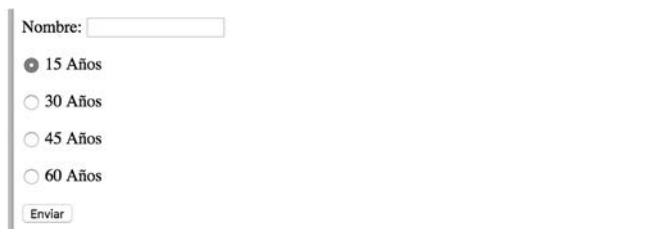
---

```
<form name="formulario" method="get" action="procesar.php">
  <p><label>Nombre: <input type="text" name="nombre"
maxlength="15"></label></p>
  <p><label><input type="radio" name="edad" value="15" checked> 15
Años</label></p>
  <p><label><input type="radio" name="edad" value="30"> 30
Años</label></p>
  <p><label><input type="radio" name="edad" value="45"> 45
Años</label></p>
  <p><label><input type="radio" name="edad" value="60"> 60
Años</label></p>
  <p><input type="submit" value="Enviar"></p>
</form>
```

---

**Listado 2-55:** Implementando el tipo `radio`

En el formulario del Listado 2-55 declaramos cuatro elementos `<input>` de tipo `radio` para ofrecer distintas edades que el usuario puede elegir. Como todos los elementos tienen el mismo nombre (`edad`), se consideran parte del mismo grupo y, por lo tanto, solo una de las opciones puede ser seleccionada a la vez. Además del atributo `name`, también implementamos un atributo booleano llamado `checked`. Este atributo le dice al navegador que seleccione el botón cuando se carga el documento, lo cual determina la edad de 15 años como el valor por defecto.



Nombre:

☒ 15 Años

☐ 30 Años

☐ 45 Años

☐ 60 Años

**Figura 2-33:** Campos de entrada de tipo `radio`

Como ya mencionamos, solo se puede seleccionar uno de estos botones a la vez. El valor asignado al atributo `value` del elemento seleccionado es el que se enviará al servidor. El tipo `checkbox` genera un tipo de entrada similar. En este caso, el usuario puede seleccionar múltiples valores haciendo clic en las casillas correspondientes.

---

```

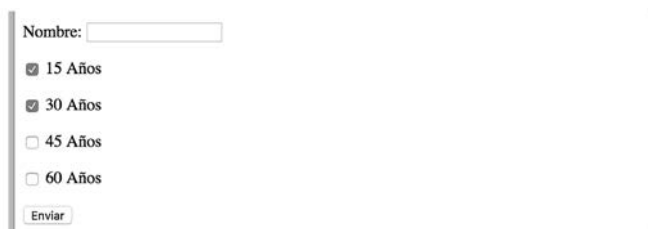
<form name="formulario" method="get" action="procesar.php">
  <p><label>Nombre: <input type="text" name="nombre"
maxlength="15"></label></p>
  <p><label><input type="checkbox" name="edad15" value="15" checked> 15
Años</label></p>
  <p><label><input type="checkbox" name="edad30" value="30" checked> 30
Años</label></p>
  <p><label><input type="checkbox" name="edad45" value="45"> 45
Años</label></p>
  <p><label><input type="checkbox" name="edad60" value="60"> 60
Años</label></p>
  <p><input type="submit" value="Enviar"></p>
</form>

```

---

**Listado 2-56: Implementando el tipo checkbox**

El tipo checkbox es similar al tipo radio, pero tenemos que asignar diferentes nombres a cada elemento porque el usuario puede seleccionar varias opciones al mismo tiempo. Cuando el usuario selecciona una o más opciones, los valores de todos esos elementos se envían al servidor. Esta clase de campo de entrada también puede incluir el atributo checked para seleccionar opciones por defecto. En nuestro ejemplo seleccionamos dos valores: 15 y 30.



**Figura 2-34: Campos de entrada de tipo checkbox**

Otro tipo de entrada que genera una herramienta visual es date. Este control le ayuda al usuario a seleccionar una fecha. Algunos navegadores lo implementan como un calendario que se muestra cada vez que el usuario hace clic en el campo. El valor enviado al servidor por este tipo de campos tiene la sintaxis año-mes-día, por lo que si queremos especificar un valor inicial o el navegador no facilita una herramienta para seleccionarlo, debemos declararlo en este formato.

---

```

<input type="date" name="fecha" value="2017-02-05">

```

---

**Listado 2-57: Implementando el tipo date**

El elemento <input> en el Listado 2-57 crea un campo de entrada con la fecha 2017-02-05 como valor por defecto. Si este elemento se muestra en un navegador que facilita un calendario para seleccionar la fecha, como Google Chrome, la fecha inicial seleccionada en el calendario será la que defina el atributo value.

Fecha: 02/05/2017

February 2017

Sun	Mon	Tue	Wed	Thu	Fri	Sat
29	30	31	1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	1	2	3	4

**Figura 2-35:** Campo de entrada de tipo date

El tipo date no es el único disponible para insertar fechas. HTML también ofrece el tipo datetime-local para seleccionar fecha y hora, el tipo week para seleccionar una semana, el tipo month para seleccionar un mes, y el tipo time para seleccionar horas y minutos. Estos tipos se crearon con diferentes propósitos y, por lo tanto, esperan valores con diferentes sintaxis. El tipo datetime-local espera un valor que representa la fecha y la hora en el formato año-mes- día horas:minutos:segundos, con la letra T separando la fecha de la hora, como en 2017-02- 05T10:35:00. El tipo week espera un valor con la sintaxis 2017- W30, donde 2017 es el año y 30 es el número de la semana. El tipo month espera una sintaxis año-mes. Y finalmente, el tipo time espera un valor que representa horas y minutos con la sintaxis horas:minutos (el carácter : es convertido a la cadena de caracteres %3ª cuando el valor se envía al servidor).



**Ejercicio 2.16 :**reemplace los elementos `<input>` en su archivo HTML por el elemento del Listado 2-57 y abra el documento en su navegador. Haga clic en el elemento. En un navegador moderno, debería ver una ventana con un calendario donde puede seleccionar una fecha. Cambie el tipo de entrada en el elemento `<input>` por cualquiera de los tipos mencionados arriba para ver las clases de herramientas que facilita el navegador para introducir estos tipos de valores. Si pulsa el botón **Enviar**, el valor seleccionado o introducido en el campo se agrega a la URL y el navegador intenta acceder al archivo procesar.php en el servidor para procesarlo. Debido a que este archivo aún no existe, el navegador devuelve un error, pero al menos podrá ver en la barra de navegación cómo se incluyen los valores en la URL. Más adelante en este capítulo estudiaremos el modo de enviar un formulario y cómo procesar los valores en el servidor.

Además de los tipos de campos de entrada disponibles para introducir fechas, existe un tipo de campo llamado color que ofrece una interfaz predefinida para seleccionar un valor. Generalmente, el valor esperado por estos campos es un número hexadecimal, como #00FF00.

---

```
<input type="color" name="micolor" value="#99BB00">
```

---

**Listado 2-58:** Implementando el tipo color

Un elemento `<input>` de tipo `color` se presenta en la pantalla como un rectángulo pintado del color determinado por defecto, seleccionado por el usuario, o definido por el atributo `value`. Cuando el usuario hace clic en este rectángulo, el navegador abre una herramienta que nos permite seleccionar un nuevo color.



**Figura 2-36:** Campo de entrada de tipo `color`



**Lo básico:** los colores se pueden expresar en varios formatos y seleccionar desde sistemas de colores diferentes. La nomenclatura más común es la hexadecimal, cuya sintaxis incluye el carácter `#` seguido por tres valores hexadecimales que representan los componentes del color (rojo, verde, y azul). Estudiaremos cómo definir colores en el Capítulo 3.

Hasta el momento hemos usado un elemento `<input>` de tipo `submit` para crear el botón que el usuario tiene que pulsar para enviar el formulario. Este tipo de entrada crea un botón estándar identificado con un título. Si queremos mejorar el diseño, podemos implementar el tipo de entrada `image` que crea un botón con una imagen. Estos tipos de campos requieren el atributo `src` con la URL del archivo que contiene la imagen que queremos usar para el botón, y pueden incluir los atributos `width` y `height` para definir su tamaño.

```
<form name="formulario" method="get" action="procesar.php">
  <p><label>Name: <input type="text" name="nombre"></label></p>
  <p><label>Age: <input type="text" name="edad"></label></p>
  <p><input type="image" src="botonenviar.png" width="100"></p>
</form>
```

**Listado 2-59:** Implementando el tipo `image` para crear un botón personalizado

El formulario del Listado 2-59 crea el botón para enviar el formulario con un elemento `<input>` de tipo `image`. El elemento carga la imagen del archivo `botonenviar.png` y la muestra en la pantalla. Cuando el usuario hace clic en la imagen, el formulario se envía del mismo modo que con los botones que hemos usado anteriormente.



**Figura 2-37:** Campo de entrada de tipo `image`

Aunque los botones creados con elementos `<input>` son probablemente más que suficientes para la mayoría de los proyectos, HTML ofrece un elemento más versátil para crear botones llamado `<button>`. Este elemento incluye el atributo `type` para determinar el tipo de botón que queremos generar. Por ejemplo, si queremos crear un botón para enviar el formulario, debemos declarar el valor `submit`.

---

```
<form name="formulario" method="get" action="procesar.php">
  <p><label>Nombre: <input type="text" name="nombre"></label></p>
  <p><label>Edad: <input type="text" name="edad"></label></p>
  <p><button type="submit">Enviar Formulario</button></p>
</form>
```

---

**Listado 2-60:** Implementando el elemento `<button>` para crear un botón



**Lo básico:** el elemento `<button>` crea un botón estándar con las mismas características que el botón creado por el elemento `<input>`. La diferencia es que el título de estos botones se define entre las etiquetas de apertura y cierre, lo cual nos permite usar otros elementos HTML e incluso imágenes para declararlo. Por esta razón, el elemento `<button>` es el que se prefiere cuando queremos personalizarlo usando estilos CSS o cuando queremos usarlo para ejecutar códigos JavaScript, como veremos en próximos capítulos.

El elemento `<input>` permite al usuario insertar o seleccionar varios tipos de valores, pero los campos de entrada generados por este elemento nos dejan introducir una sola línea de texto. HTML ofrece el elemento `<textarea>` para insertar múltiples líneas de texto. El elemento está compuesto por etiquetas de apertura y cierre, y puede incluir los atributos `rows` y `cols` para definir el ancho y la altura del área en caracteres.

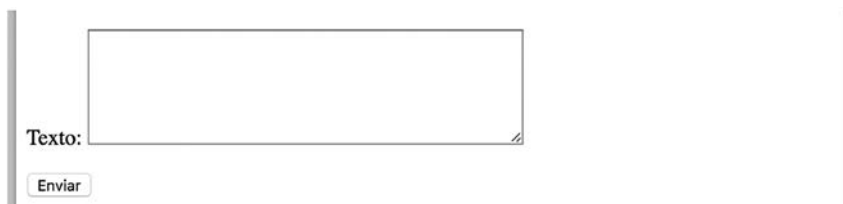
---

```
<form name="formulario" method="get" action="procesar.php">
  <p><label>Texto: <textarea name="texto" cols="50"
rows="6"></textarea></label></p>
  <p><input type="submit" value="Enviar"></p>
</form>
```

---

**Listado 2-61:** Implementando el elemento `<textarea>`

En la ventana del navegador, el elemento `<textarea>` se representa por un recuadro vacío del tamaño determinado por sus atributos o los estilos por defecto. El texto insertado queda limitado por el ancho del área, pero se puede extender verticalmente todo lo que sea necesario. Si el área no es lo suficientemente larga para contener el texto completo, el navegador muestra barras laterales con las que el usuario puede desplazar el contenido.

A screenshot of a web browser window. On the left, there is a vertical scrollbar. The main content area shows a form with the label "Texto:" followed by a large, empty rectangular text area. Below the text area is a button labeled "Enviar". On the right, there is another vertical scrollbar.

**Figura 2-38:** El elemento `<textarea>`



**Ejercicio 2.17 :**reemplace el formulario en su archivo HTML por el formulario del Listado 2-61 y abra el documento en su navegador. Escriba varias líneas de texto para ver cómo se distribuye el texto dentro del área.



**Lo básico:** si necesita declarar un valor inicial para el elemento `<textarea>`, puede insertarlo entre las etiquetas de apertura y cierre.

Además de los tipos de campos de entrada radio y checkbox estudiados anteriormente, HTML ofrece el elemento `<select>` para presentar una lista de valores al usuario. Cuando el usuario hace clic en este elemento, la lista se muestra en una ventana desplegable, y luego el valor seleccionado por el usuario se inserta en el campo. Debido a que el elemento `<select>` no genera un campo de entrada, el usuario no puede insertar valores distintos de los incluidos en la lista.

El elemento `<select>` trabaja junto con el elemento `<option>` para definir las opciones. El elemento `<select>` debe incluir el atributo `name` para identificar el valor, y cada elemento `<option>` debe incluir el atributo `value` para definir el valor que representa.

```
<form name="formulario" method="get" action="procesar.php">
  <p>
    <label for="listado">Libros: </label>
    <select name="libro" id="listado">
      <option value="1">IT</option>
      <option value="2">Carrie</option>
      <option value="3">El Resplandor</option>
      <option value="4">Misery</option>
    </select>
  </p>
  <p><input type="submit" value="Enviar"></p>
</form>
```

**Listado 2-62:** Implementando el elemento `<select>`



En el ejemplo del Listado 2-62, incluimos cuatro opciones, una para cada libro que queremos mostrar en la lista. Los valores de las opciones son definidos desde 1 a 4. Por consiguiente, cada vez que el usuario selecciona un libro y envía el formulario, el número correspondiente a ese libro se envía al servidor con el identificador "libro" (el valor del atributo name).



**Figura 2-39:** Seleccionando un valor con el elemento `<select>`

Otra manera de crear una lista predefinida es con el elemento `<datalist>`. Este elemento define una lista de ítems que, con la ayuda del atributo `list`, se puede usar como sugerencia en un campo de entrada. Al igual que las opciones para el elemento `<select>`, las opciones para este elemento se definen por elementos `<option>`, pero en este caso deben incluir el atributo `label` con una descripción del valor. El siguiente ejemplo crea un formulario con un campo de entrada para insertar un número telefónico. El código incluye un elemento `<datalist>` con dos elementos `<option>` que definen los valores que queremos sugerir al usuario.

```
<form name="formulario" method="get" action="procesar.php">
  <datalist id="datos">
    <option value="123123123" label="Teléfono 1">
    <option value="456456456" label="Teléfono 2">
  </datalist>
  <p><label>Teléfono: <input type="tel" name="telefono"
list="datos"></label></p>
  <p><input type="submit" value="Enviar"></p>
</form>
```

**Listado 2-63:** Sugiriendo una lista de valores con el elemento `<datalist>`

Para conectar un campo de entrada con un elemento `<datalist>`, tenemos que incluir el atributo `list` en el elemento `<input>` con el mismo valor que usamos para identificar el elemento `<datalist>`. Para este propósito, en el formulario del Listado 2-63 incluimos el atributo `id` en el elemento `<datalist>` con el valor "datos" y luego asignamos este mismo valor al atributo `list` del elemento `<input>`. En consecuencia, el campo de entrada muestra una flecha que despliega una lista de valores predefinidos que el usuario puede seleccionar para completar el formulario.



**Figura 2-40:** Valores predefinidos con el elemento `<datalist>`



**Ejercicio 2.18:**reemplace el formulario en su archivo HTML con el formulario del Listado 2-63. Abra el documento en su navegador y haga clic en la flecha del lado derecho del campo. Debería ver algo similar a lo que se muestra en la Figura 2-40.

HTML incluye otros dos elementos que podemos usar en un formulario: `<progress>` y `<meter>`, que no se consideran elementos de formulario, pero debido a que representan medidas, son realmente útiles cuando nuestro formulario produce esta clase de información.

El elemento `<progress>` se usa para informar del progreso en la ejecución de una tarea. Requiere dos atributos que determinan el valor actual y el máximo. El atributo `value` indica el progreso logrado hasta el momento, y el atributo `max` declara el valor que necesitamos alcanzar para dar por finalizada la tarea.

---

```
<progress value="30" max="100">0%</progress>
```

---

**Listado 2-64:** Implementando el elemento `<progress>`

Los navegadores representan este elemento con una barra de dos colores. Por defecto, la porción de la barra que representa el progreso se muestra en color azul. Si el navegador no reconoce el elemento, el valor entre las etiquetas de apertura y cierre se muestra en su lugar.



**Figura 2-41:** Barra de progreso

Al igual que el elemento `<progress>`, el elemento `<meter>` se usa para mostrar una escala, pero no representa progreso. Su propósito es representar un rango de valores predefinidos (por ejemplo, el espacio ocupado en un disco duro). Este elemento cuenta con varios atributos asociados. Los atributos `min` y `max` determinan los límites del rango, `value` determina el valor medido, y `low`, `high` y `optimum` se usan para segmentar el rango en secciones diferenciadas y declarar la posición óptima.

---

```
<meter value="60" min="0" max="100" low="40" high="80" optimum="100">60</meter>
```

---

**Listado 2-65:** Implementando el elemento `<meter>`

El código del Listado 2-65 genera una barra en la pantalla que muestra un nivel de 60 en una escala de 0 a 100 (de acuerdo con los valores declarados por los atributos value, min y max). El color de la barra generada por el elemento depende de los niveles determinados por los atributos low, high y optimum. Debido a que el valor actual en nuestro ejemplo se encuentra entre los valores de los atributos low y high, la barra se muestra en color amarillo.



**Figura 2-42:** Barra creada con el elemento `<meter>`

## Enviando el formulario

Al enviar un formulario, los datos introducidos se envían al servidor usando la sintaxis nombre/valor para cada elemento, donde nombre es el valor asignado al atributo name del elemento y valor es el valor que introduce el usuario. Por ejemplo, si insertamos el texto "Roberto" en un campo de entrada con el nombre "minombre", el par nombre/valor que se envía al servidor será minombre/Roberto.

Los navegadores usan dos métodos para enviar esta información: GET y POST. El método se declara asignando los valores GET o POST al atributo method del elemento <form>, como hemos hecho en ejemplos anteriores, aunque el método a usar depende del tipo de información gestionada por el formulario. Como mencionamos en el Capítulo 1, los navegadores se comunican con el servidor usando un protocolo llamado HTTP. Cuando el navegador quiere acceder a un documento, envía una solicitud HTTP al servidor. Una solicitud HTTP es un mensaje que le dice al servidor cuál es el recurso al que el navegador quiere acceder y lo que quiere hacer con el mismo (descargar el documento, procesar información, etc.). El mensaje está compuesto por la URL del recurso, los datos asociados con la solicitud, como la fecha y el idioma, y un cuerpo con información adicional. Los pares nombre/valor producidos por un formulario se envían al servidor dentro de estas solicitudes HTTP. Si el método se ha declarado como GET, los pares nombre/valor se agregan al final de la URL, pero si el método se ha declarado como POST, los valores se incluyen en el cuerpo de la solicitud. Esto significa que la información enviada con el método GET es visible para el usuario (el usuario puede ver la URL con todos los pares nombre/valor en la barra de navegación del navegador), pero la información enviada con el método POST se oculta dentro de la solicitud. En consecuencia, si la información es sensible o privada, debemos usar el método POST, pero si la información no es sensible, como valores de búsqueda insertados por el usuario, podemos usar el método GET.

Así mismo, tenemos que considerar que el método POST se puede usar para enviar una cantidad ilimitada de información, pero el método GET tiene que adaptarse a las limitaciones presentadas por las URL. Esto se debe a que el largo de una URL es limitado. Si la información insertada en el formulario es muy extensa, se podría perder.

El siguiente ejemplo presenta un formulario con un único campo de entrada para ilustrar cómo funciona este proceso. El elemento <input> se identifica con el nombre "val", y el método utilizado para enviar la información se declara como GET, lo que significa que el valor insertado por el usuario se agregará a la URL.

---

```

<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="utf-8">
  <title>Formularios</title>
</head>
<body>
  <section>
    <form name="formulario" method="get" action="procesar.php">
      <p><input type="text" name="val"></p>
      <p><input type="submit" value="Enviar"></p>
    </form>
  </section>
</body>
</html>

```

---

**Listado 2-66:** Formularios con el método GET

En el ejemplo del Listado 2-66, un archivo llamado procesar.php se declara a cargo de procesar la información. Cuando el usuario pulsa el botón Enviar, el navegador crea una solicitud HTTP que incluye la URL que apunta a este archivo. Debido a que el método del formulario se ha declarado como GET, el nombre del campo de entrada y el valor insertado por el usuario se agregan a la URL, tal como se muestra a continuación.

**www.ejemplo.com/procesar.php?val=10**

---

Dominio

---

Parámetro

**Figura 2-43:** Par nombre/valor en la URL

Cuando la información se envía con el método GET, los pares nombre/valor se agregan al final de la URL separados por el carácter =, y el primer par es precedido por el carácter ?. Si existe más de un par de valores, los restantes se agregan a la URL separados por el carácter &, como en [www.ejemplo.com/procesar.php?val1=10&val2=20](http://www.ejemplo.com/procesar.php?val1=10&val2=20).

Al otro lado, el servidor recibe esta solicitud, lee la URL, extrae los valores y ejecuta el código en el archivo procesar.php. Este código debe procesar la información recibida y producir una respuesta. La forma en la que se realiza esta tarea depende del lenguaje de programación que utilizamos y lo que queremos lograr. Por ejemplo, para leer los valores enviados con el método GET, PHP los ofrece en un listado llamado \$\_GET. La sintaxis requerida para obtener el valor incluye su nombre entre corchetes.

---

```

<?php
  print('El valor es: ' . $_GET['val']);
?>

```

---

**Listado 2-67:** Procesando los datos en el servidor con PHP (procesar.php)

El ejemplo del Listado 2-67 muestra cómo procesar valores con PHP. Las etiquetas `<?php` y `?>` indican al servidor que este es código PHP y que tiene que ser ejecutado como tal. El código puede ser extenso o estar compuesto por solo unas pocas instrucciones, dependiendo de lo que necesitamos. Nuestro ejemplo incluye una sola instrucción para ilustrar cómo se procesa la información. Esta instrucción, llamada `print()`, toma los valores entre paréntesis y los incluye en un archivo que se va a devolver al navegador como respuesta. En este caso, agregamos el valor recibido por medio de la solicitud al texto "El valor es: ". Si enviamos el valor 10, como en el ejemplo de la Figura 2-43, el servidor genera un archivo con el texto "El valor es: 10" y lo envía de regreso al navegador.



**Lo básico:** el código PHP de nuestro ejemplo utiliza el nombre `$_GET` para capturar la información recibida desde el navegador porque el método del formulario se ha declarado como GET, pero si cambiamos el método a POST, debemos utilizar el nombre `$_POST`.

El archivo producido por el servidor a través de un código PHP es un archivo HTML. Por propósitos didácticos, no incluimos ningún elemento HTML en el ejemplo del Listado 2-67, pero siempre deberíamos generar un documento HTML válido en respuesta. Existen varias maneras de definir un documento en PHP. La más simple es crear el documento HTML como lo hemos hecho anteriormente pero dentro de un archivo PHP, e incluir el código PHP donde queremos mostrar el resultado. Por ejemplo, el siguiente ejemplo inserta código PHP dentro de un elemento `<p>`.

Cuando el servidor abre el archivo `procesar.php` con este documento, ejecuta el código PHP, inserta el resultado dentro de las etiquetas `<p>`, y devuelve el archivo al servidor. El resultado es el mismo que si hubiéramos creado un documento estático con el texto "El valor es: 10", pero el texto se genera dinámicamente en el servidor con los valores recibidos desde el formulario.

---

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="utf-8">
  <title>Respuesta</title>
</head>
<body>
  <section>
    <p>
      <?php
        print('El valor es: ' . $_GET['val']);
      ?>
    </p>
  </section>
</body>
</html>
```

---

**Listado 2-68:** Generando un documento HTML que incluye código PHP (`procesar.php`)



**Ejercicio 2.19** :para probar este ejemplo, tiene que subir los archivos a un servidor que pueda ejecutar código PHP. Si no posee una cuenta de alojamiento con estas características, puede instalar un servidor en su ordenador con paquetes como MAMP, introducido en el Capítulo 1 (si no sabe cómo usar un servidor, no se preocupe, este procedimiento no es necesario para realizar la mayoría de los ejemplos de este libro). Cree un nuevo archivo HTML con el código del Listado 2-66, y un archivo llamado procesar.php con el código del Listado 2-67 o 2-68. Suba los archivos al servidor y abra el documento en su navegador. Inserte un valor dentro del campo de entrada y pulse el botón Enviar. El navegador debería mostrar una nueva página que contenga el texto que comienza con la cadena de caracteres "El valor es: " y termina con el valor que ha insertado en el formulario.



**IMPORTANTE:** este es simplemente un ejemplo de cómo se procesa la información y cómo el navegador transmite los datos introducidos en el formulario al servidor. Estudiaremos otros ejemplos parecidos en próximos capítulos, pero no es el propósito de este libro enseñar cómo programar en PHP u otro lenguaje de programación de servidor. Para aprender más sobre PHP, vaya a nuestro sitio web y visite las secciones Enlaces y Vídeos.

## Atributos globales

HTML define atributos globales que son exclusivos de elementos de formulario. Los siguientes son los más utilizados.

**disabled**—Este es un atributo booleano que desactiva el elemento. Cuando el atributo está presente, el usuario no puede introducir valores o interactuar con el elemento.

**readonly**—Este atributo indica que el valor del elemento no se puede modificar.

**placeholder**—Este atributo muestra un texto en el fondo del elemento que indica al usuario el valor que debe introducir.

**autocomplete**—Este atributo activa o desactiva la función de autocompletar. Los valores disponibles son **on** y **off**.

**novalidate**—Este es un atributo booleano para el elemento `<form>` que indica que el formulario no debería ser validado.

**formnovalidate**—Este es un atributo booleano para los elementos `<button>` e `<input>` de tipo **submit** e **image** que indica que el formulario al que pertenecen no debería ser validado.

**required**—Este es un atributo booleano que indica al navegador que el usuario debe seleccionar o insertar un valor en el elemento para validar el formulario.

**multiple**—Este es un atributo booleano que indica al navegador que se pueden insertar múltiples valores en el campo (se aplica a elementos `<input>` de tipo **email** y **file**).

**autofocus**—Este atributo booleano solicita al navegador que mueva el foco al elemento tan pronto como se carga el documento.

**pattern**—Este atributo define una expresión regular que el navegador debe usar para validar el valor insertado en el campo.

**form**—Este atributo asocia el elemento con un formulario. Se usa para conectar un elemento con un formulario cuando el elemento no se define entre las etiquetas `<form>`. El valor asignado a este atributo debe ser el mismo asignado al atributo **id** del elemento `<form>`.

**spellcheck**—Este atributo solicita al navegador que compruebe la ortografía y gramática del valor introducido en el campo. Los valores disponibles son **true** (verdadero) y **false** (falso).

Los atributos **disabled** y **readonly** tienen un propósito similar, no permitir al usuario interactuar con el elemento, pero se aplican en diferentes circunstancias. El atributo **disabled** normalmente se implementa cuando queremos mostrar al usuario que el elemento puede estar disponible en otras condiciones, como cuando el control no es aplicable en el país del usuario, por ejemplo. Por otro lado, el atributo **readonly** se implementa cuando solo existe un valor posible y no queremos que el usuario lo cambie.



Por ejemplo, en el siguiente formulario, el usuario no puede introducir la edad.

---

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="utf-8">
  <title>Formularios</title>
</head>
<body>
  <section>
    <form name="formulario" method="get" action="procesar.php">
      <p><label>Nombre: <input type="text" name="nombre"></label></p>
      <p><label>Edad: <input type="text" name="edad" disabled></label></p>
      <p><input type="submit" value="Enviar"></p>
    </form>
  </section>
</body>
</html>
```

---

**Listado 2-69:** Implementando el atributo `disabled`

Los elementos afectados por los atributos `disabled` y `readonly` se muestran en colores más claros para advertir al usuario de que no son controles normales. Otro atributo que afecta la apariencia de un elemento es `placeholder`. Este se usa en campos de entrada para ofrecer una pista (una palabra o frase) que ayude al usuario a introducir el valor correcto. El siguiente ejemplo inserta esta ayuda en un campo de búsqueda.

---

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="utf-8">
  <title>Formularios</title>
</head>
<body>
  <section>
    <form name="formulario" method="get" action="procesar.php">
      <p><label>Buscar: <input type="search" name="buscar"
placeholder="Término a buscar"></label></p>
      <p><input type="submit" value="Buscar"></p>
    </form>
  </section>
</body>
</html>
```

---

**Listado 2-70:** Implementando el atributo `placeholder`

El valor de este atributo lo muestran los navegadores dentro del campo hasta que el usuario inserta un valor.



**Figura 2-44:** Campo de entrada con un mensaje de ayuda

Una de las características de los formularios es que tienen la capacidad de validar los datos introducidos. Por defecto, los formularios validan los datos a menos que el atributo `novalidate` sea declarado. Este atributo booleano es específico de elementos `<form>`. Cuando es incluido, el formulario se envía sin validar. La presencia de este atributo afecta al formulario de forma permanente, pero a veces el proceso de validación es requerido solo en ciertas circunstancias. Por ejemplo, cuando la información insertada debe ser grabada para permitir al usuario continuar con el trabajo más adelante. En casos como este, podemos implementar el atributo `formnovalidate`. Este atributo está disponible para los elementos que crean los botones para enviar el formulario. Cuando los datos se envían con un botón que contiene este atributo, el formulario no es validado.

---

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="utf-8">
  <title>Formularios</title>
</head>
<body>
  <section>
    <form name="formulario" method="get" action="procesar.php">
      <p><label>Correo: <input type="email" name="correo"></label></p>
      <p>
        <input type="submit" value="Enviar">
        <input type="submit" value="Grabar" formnovalidate>
      </p>
    </form>
  </section>
</body>
</html>
```

---

**Listado 2-71:** Enviando un formulario sin validar con el atributo `formnovalidate`

En el ejemplo del Listado 2-71, el formulario será validado en circunstancias normales, pero incluimos un segundo botón con el atributo `formnovalidate` para poder enviar el formulario sin pasar por el proceso de validación. El botón `Enviar` requiere que el usuario introduzca una cuenta de correo válida, pero el botón `Grabar` no incluye este requisito.

Cuando usamos el tipo email para recibir una cuenta de correo, como en el ejemplo anterior, el navegador controla si el valor introducido es una cuenta de correo, pero valida el campo cuando se encuentra vacío. Esto se debe a que el campo no es obligatorio. HTML ofrece el atributo required para cambiar esta condición. Cuando se incluye el atributo required, el campo solo será válido si el usuario introduce un valor y este valor cumple con los requisitos de su tipo. El siguiente ejemplo implementa este atributo para forzar al usuario a introducir un correo electrónico.

---

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="utf-8">
  <title>Formularios</title>
</head>
<body>
  <section>
    <form name="formulario" method="get" action="procesar.php">
      <p><label>Correo: <input type="email" name="correo"
required></label></p>
      <p>
        <input type="submit" value="Enviar">
        <input type="submit" value="Grabar" formnovalidate>
      </p>
    </form>
  </section>
</body>
</html>
```

---

**Listado 2-72:** Declarando una entrada email como campo requerido



**Ejercicio 2.20** : cree un archivo HTML con el código del Listado 2-72. Abra el documento en su navegador y pulse el botón enviar. Debería recibir un mensaje de error en el campo **correo** indicando que debe insertar un valor. Pulse el botón **Grabar**. Como este botón incluye el atributo **formnovalidate**, el error ya no se muestra y se envía el formulario.

Otro atributo que se usa para validación es pattern. Algunos tipos de campos de entrada validan cadenas de caracteres específicas, pero no pueden hacer nada cuando el valor no es estándar, como en el caso de los códigos postales. No existe un tipo de campo predeterminado para esta clase de valores. El atributo pattern nos permite crear un filtro personalizado usando expresiones regulares.

Las expresiones regulares son textos compuestos por una serie de caracteres que definen un patrón de concordancia. Por ejemplo, los caracteres 0-9 determinan que solo se aceptan los números entre 0 y 9. Utilizando esta clase de expresiones, podemos crear un filtro personalizado para validar cualquier valor que necesitemos. El siguiente formulario incluye un campo de entrada que solo acepta números con cinco dígitos.

---

```

<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="utf-8">
  <title>Formularios</title>
</head>
<body>
  <section>
    <form name="formulario" method="get" action="procesar.php">
      <p><label>Código Postal: <input pattern="[0-9]{5}" name="cp"
title="Inserte su código postal"></label></p>
      <p><input type="submit" value="Enviar"></p>
    </form>
  </section>
</body>
</html>

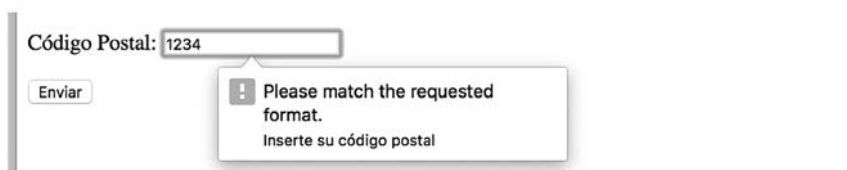
```

---

**Listado 2-73:** Personalizando campos de entrada con el atributo `pattern`

El elemento `<input>` en el Listado 2-73 incluye el atributo `pattern` con el valor `[0-9]{5}`. Esta expresión regular determina que el valor debe tener exactamente cinco caracteres y que esos caracteres deben ser números entre 0 y 9. En consecuencia, solo podemos insertar números de cinco dígitos; cualquier otro carácter o tamaño devolverá un error.

Estos tipos de entrada también pueden incluir el atributo `title` para explicar al usuario cuál es el valor esperado. Este mensaje complementa el mensaje de error estándar que muestra el navegador, tal como ilustra la Figura 2-45.



**Figura 2-45:** Error en un campo de entrada con un patrón personalizado



**IMPORTANTE:** no es el propósito de este libro enseñar cómo trabajar con expresiones regulares. Para más información, visite nuestro sitio web y siga los enlaces de este capítulo.

Los tipos de entrada `email` y `file` solo permiten al usuario introducir un valor a la vez. Si queremos permitir la inserción de múltiples valores, tenemos que incluir el atributo `multiple`. Con este atributo, el usuario puede insertar todos los valores que quiera separados por coma. El siguiente ejemplo incluye un campo de entrada que acepta múltiples direcciones de correo.

---

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="utf-8">
  <title>Formularios</title>
</head>
<body>
  <section>
    <form name="formulario" method="get" action="procesar.php">
      <p><label>Correo: <input type="email" name="correo"
multiple></label></p>
      <p><input type="submit" value="Enviar"></p>
    </form>
  </section>
</body>
</html>
```

---

**Listado 2-74:** Declarando un campo de entrada email como campo múltiple

Además de los atributos de validación, existen otros atributos que pueden ayudar al usuario a decidir qué valores introducir. Por ejemplo, el atributo autocomplete activa una herramienta que le sugiere al usuario qué introducir según los valores insertados previamente. Los valores disponibles para este atributo son on y off (activar y desactivar la herramienta). Este atributo también se puede implementar en el elemento <form> para que afecte a todos los elementos del formulario. El siguiente formulario desactiva las sugerencias para una búsqueda.

---

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="utf-8">
  <title>Formularios</title>
</head>
<body>
  <section>
    <form name="formulario" method="get" action="procesar.php">
      <p><label>Buscar: <input type="search" name="buscar"
autocomplete="off"></label></p>
      <p><input type="submit" value="Buscar"></p>
    </form>
  </section>
</body>
</html>
```

---

**Listado 2-75:** Implementando el atributo autocomplete



**Ejercicio 2.21** :cree un archivo HTML con el código del Listado 2-75. Abra el documento en su navegador. Inserte un valor y presione el botón **Enviar**. Repita el proceso. El navegador no debería sugerir ningún valor a insertar. Cambie el valor del atributo **autocomplete** a **on** y repita el proceso. En esta oportunidad, cada vez que introduzca un valor, el navegador le mostrará un listado con los valores insertados previamente que comienzan con los mismos caracteres.

Otro atributo que puede ayudar al usuario a decidir qué insertar es autofocus. En este caso, el atributo establece el foco en el elemento cuando se carga el documento, sugiriendo al usuario qué valor insertar primero. El siguiente ejemplo incluye dos campos de entrada para insertar el nombre y la edad del usuario, pero el campo para la edad incluye el atributo autofocus y, por lo tanto, el navegador posicionará el cursor en este campo por defecto.

---

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="utf-8">
  <title>Formularios</title>
</head>
<body>
  <section>
    <form name="formulario" method="get" action="procesar.php">
      <p><label>Nombre: <input type="text" name="nombre"></label></p>
      <p><label>Edad: <input type="text" name="edad" autofocus></label></p>
      <p><input type="submit" value="Enviar"></p>
    </form>
  </section>
</body>
</html>
```

---

**Listado 2-76:** *Implementando el atributo autofocus*

Otra herramienta que pueden activar automáticamente los navegadores es el control de ortografía. A pesar de la utilidad de esta herramienta y de que los usuarios esperan casi siempre tenerla a su disposición, puede resultar inapropiada en ciertas circunstancias. La herramienta se activa por defecto, pero podemos incluir el atributo spellcheck con el valor false para desactivarla, como hacemos en el siguiente ejemplo.

---

```

<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="utf-8">
  <title>Formularios</title>
</head>
<body>
  <section>
    <form name="formulario" method="get" action="procesar.php">
      <p><label>Texto: <textarea name="texto" cols="50" rows="6"
spellcheck="false"></textarea></label></p>
      <p><input type="submit" value="Enviar"></p>
    </form>
  </section>
</body>
</html>

```

---

**Listado 2-77:** Desactivando el control de ortografía

Finalmente, existe otro atributo útil que podemos implementar para declarar como parte del formulario elementos que no se han incluido entre las etiquetas <form>. Por ejemplo, si tenemos un campo de entrada en el área de navegación, pero queremos que el elemento se envíe con el formulario definido en el área central del documento, podemos conectar este elemento con el formulario usando el atributo form (el valor de este atributo debe coincidir con el valor del atributo id asignado al elemento <form>).

---

```

<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="utf-8">
  <title>Formularios</title>
</head>
<body>
  <nav>
    <p><input type="search" name="buscar" form="formulario"></p>
  </nav>
  <section>
    <form name="formulario" id="formulario" method="get"
action="procesar.php">
      <p><input type="submit" value="Enviar"></p>
    </form>
  </section>
</body>
</html>

```

---

**Listado 2-78:** Declarando elementos de formulario en una parte diferente del documento



**IMPORTANTE:** los formularios son extremadamente útiles en el desarrollo web, y se requieren en la mayoría de los sitios y aplicaciones web modernas. Los ejemplos de este capítulo se han creado con propósitos didácticos y se han simplificado al máximo para que pueda probar los elementos y controles disponibles. En próximos capítulos, estudiaremos cómo implementar formularios en situaciones más prácticas y expandirlos con CSS y JavaScript.



## Listado de ejercicios

Ejercicio 2.1_ <title>	6
Ejercicio 2.2_ favicon	8
Ejercicio 2.3_ estructura errores	8
Ejercicio 2.4_ estructura <header>	12
Ejercicio 2.5_ texto <h1>	21
Ejercicio 2.6_ enlace externo <a>	28
Ejercicio 2.7_ enlace interno <a>	30
Ejercicio 2.8_ enlace mailto <a>	31
Ejercicio 2.9_ imagen <img>	35
Ejercicio 2.10_ listado <ul> <ol>	38
Ejercicio 2.11_ <span> translate	44
Ejercicio 2.12_ <span> contenteditable	45
Ejercicio 2.13_ formulario <form>	47
Ejercicio 2.14_ <input>	51
Ejercicio 2.15_ <input> atributos (1)	56
Ejercicio 2.16_ <input> atributos (2)	59
Ejercicio 2.17_ <textarea>	62
Ejercicio 2.18_ <datalist>	63
Ejercicio 2.19_ envio de formulario	69
Ejercicio 2.20_ formnovalidate	73
Ejercicio 2.21_ autocomplete	76
Ejercicio 2.22_ replicar documento de word	79
Ejercicio 2.23.1_ listas anidadas (1)	80
Ejercicio 2.23.2_ listas anidadas (2)	80
Ejercicio 2.24.1_ tabla (1)	81
Ejercicio 2.24.2_ tabla (2)	82