

2.3 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.

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>**



Hágalo usted mismo: 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.



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



Hágalo usted mismo: 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 lo 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"></label>
</p>
<p>
    <label for="edad">Edad: </label>
    <input type="text" name="edad" id="edad"></label>
</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.

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.

The figure shows a vertical form with five input fields. From top to bottom: 'Nombre:' followed by a text input field; 'Edad:' followed by a number input field with up and down arrow buttons on its right; 'Correo:' followed by a text input field; 'Teléfono:' followed by a text input field; and 'Sitio Web:' followed by a text input field. At the bottom of the form is a blue rectangular button labeled '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



Hágalo usted mismo: 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.

Nombre:

15 Años
 30 Años
 45 Años
 60 Años

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**.



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).



Hágalo usted mismo: 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