

Formularios



Hasta ahora, la única forma que hemos visto de interactuar con una web son los enlaces, haciendo clic en ellos. Pero existe otra opción más potente para que el usuario interactúe con una web: los formularios. Son una herramienta muy útil para enviar información a un servidor. Cuando nos registramos en una web, rellenamos una serie de datos (e-mail, contraseña, etc.) mediante un formulario. Así pues, un formulario contiene una serie de elementos (llamados normalmente "controles" o "campos") que colocamos en una página web para que el usuario pueda introducir información. Veremos a continuación qué tipos de elementos puede tener un formulario y cómo se colocan en la página.

1. Estructura básica de un formulario

Todo el contenido de un formulario va encerrado entre una pareja de etiquetas `<form>`. Esta etiqueta tiene, entre otros, los siguientes atributos:

- `action`: sería el equivalente al *href* de los enlaces. Indica a qué página queremos ir cuando rellenemos el formulario y lo enviemos. Podemos utilizar, como en los enlaces, una ruta relativa o absoluta a la página. Lo normal es que la página a la que ir en el *action* de un formulario sea una página escrita en lenguaje de servidor (PHP, JSP, ASP... según el lenguaje que hayamos elegido), ya que normalmente el servidor debe recoger la información que envía el usuario en el formulario, procesarla y hacer algo con ella. Por ejemplo, si nos estamos registrando en la web, el servidor deberá procesar nuestros datos y darnos de alta en la base de datos de usuarios de la web.
- `method`: indica la forma en que se envían los datos al servidor. Puede valer *get* (los datos son visibles en la barra de direcciones del navegador cuando los enviamos) o *post* (los datos no son visibles desde la barra de direcciones, y por tanto es algo más seguro).

```
<form action="mipagina.php" method="post">
  ... Contenido del formulario
</form>
```

- `autocomplete`: puede tomar los valores *on* y *off*, siendo *on* el valor por defecto. Si está activado, al rellenar cada campo del formulario se mostrará un listado de términos que hemos escrito recientemente, para ayudarnos a completar la información en el caso de que queramos que sea la misma. Por ejemplo, en campos de tipo *e-mail* se mostrará una lista con los últimos e-mails que hayamos escrito en formularios. Este mismo atributo también existe en los controles internos del formulario, de modo que podemos habilitar el autocompletado para el formulario en general, y deshabilitarlo para ciertos controles.

2. Elementos básicos de un formulario

Dentro de las etiquetas *form* que engloban al formulario, podemos poner diferentes tipos de controles. Los más habituales son:

- **Cuadros de texto:** para introducir en una línea una información corta y concreta, como por ejemplo un e-mail o un password.
- **Áreas de texto:** para introducir textos largos, de más de una línea. Por ejemplo, el contenido de una noticia.
- **Checkboxes:** casillas de verificación que se marcan y desmarcan
- **Botones de radio o radio buttons:** conjunto de botones u opciones entre los que sólo puede haber uno seleccionado.
- **Listas:** desplegables o fijas
- **Botones:** para enviar el formulario, o borrar los datos del mismo, por ejemplo
- **Etiquetas:** dan información de otros controles, indicando para qué sirven.

La mayor parte de estos controles (cuadros de texto, *checkboxes*, botones de radio, botones normales...) se añaden con la etiqueta `input`. Otros controles necesitan de otras etiquetas. Veremos todas ellas a continuación.

2.1. La etiqueta *input*

La etiqueta `input` se utiliza para definir distintos tipos de controles, dependiendo de lo que valga su atributo `type`. Este atributo puede valer:

- `text`: para definir cuadros de texto simples (de una sola línea).
- `password`: para definir cuadros de texto enmascarados (para contraseñas)
- `file`: para definir controles para subir archivos al servidor. Aparecerá un cuadro de texto y un botón para que, al pinchar en él, podamos elegir el archivo. Estos controles no son de mucha utilidad sin un lenguaje de servidor que les ayude a completar la tarea, como PHP, JSP o ASP.NET. Se puede añadir el atributo `multiple` en el caso de que queramos poder adjuntar varios archivos.
- `checkbox`: para definir casillas de verificación o *checkboxes*
- `radio`: para definir botones de radio.
- `submit`: para definir el botón que servirá para enviar el formulario al pulsarlo
- `reset`: para el botón que servirá para borrar el contenido del formulario al pulsarlo
- `button`: para definir otro botón cualquiera, con otra funcionalidad. Esta última opción no es demasiado útil de momento, si no se combina con lenguajes de cliente como Javascript.
- `hidden`: sirve para definir campos ocultos, que almacenan información junto con el formulario pero no la muestran al usuario. Aparentemente no tienen utilidad, pero combinado con lenguajes de servidor como PHP, pueden ser de mucha ayuda.

A partir de HTML 5 se añadieron otros tipos (*types*) válidos que permitían definir controles específicos. Podemos destacar los siguientes:

- `email`: para cuadros de texto donde se requiera introducir una dirección de correo electrónico
- `number`: para cuadros que exijan un dato numérico (real o entero)

- `date`, `time` o `datetime` : para cuadros que exijan poner una fecha, hora o fecha y hora conjunta, respectivamente
- `color` : para controles que permiten elegir un color de una paleta de colores
- `search` : para cuadros de búsqueda. En algunos navegadores se habilita estos controles con un aspa para borrar el contenido del cuadro.
- `url` : para introducir direcciones de internet.
- `range` : para controles de tipo *slider*, con una barra que podemos mover entre unos límites. Aunque, desafortunadamente, no se muestra ninguna información visual del valor actualmente seleccionado (se debería utilizar algo de código JavaScript y elementos adicionales para poder ver este valor)

Una vez elegido el tipo de control, la etiqueta *input* tiene otra serie de atributos que debemos tener en cuenta (muchos de estos atributos también están presentes en otros tipos de controles):

- `name` : sirve para dar un nombre identificativo al control. Nos será muy útil para identificarlo luego en los lenguajes de cliente o de servidor (JavaScript, PHP...). Por ejemplo todos los botones de radio que queramos agrupar en funcionalidad, deben tener la misma etiqueta *name*.
- `id` : el identificador del control, al igual que podemos usar este atributo en otras etiquetas HTML. Se utiliza en CSS, y también para algunas cosas con JavaScript. A menudo se utiliza junto a *name*, dando a los dos el mismo nombre o valor.
- `value` : sirve para dar un valor inicial con el que rellenar el control (hasta que el usuario se lo cambie)
- `size` : sirve para indicar el tamaño del control (en caracteres si es un cuadro de texto, o en píxeles para el resto)
- `maxlength` : sirve para indicar el máximo número de caracteres que puede aceptar el control (en el caso de cuadros de texto).
- `checked` : sirve para indicar, si es un checkbox o botón de radio, que está marcado. Es un valor de *verdadero o falso*, por lo que, si no se pone, asumimos que el control no está marcado, y si se pone sí lo está
- `disabled` : sirve para deshabilitar un control, y que el usuario no pueda modificar su valor. Se usa sobre todo para dar información al usuario, pero sin que éste la pueda modificar. Es un valor de *verdadero o falso*, como el anterior, y se utiliza de la misma forma. Si un campo está *disabled* no se envía cuando se envía el formulario.
- `readonly` : similar al anterior, sirve para que un control sea de sólo lectura y el usuario no pueda modificar su valor. A diferencia del anterior, el contenido sí se envía con el resto del formulario.
- `placeholder` : muestra un texto informativo dentro del cuadro de texto, que normalmente indica al usuario el tipo de información que debe introducir en dicho cuadro.

Veamos un ejemplo: el siguiente formulario muestra un cuadro de texto para poner nuestro nombre, otro que admite hasta 50 caracteres para un e-mail, una casilla de verificación para ver si el usuario está casado, dos botones de radio para elegir su idioma preferido, y un botón para enviar el formulario.

```
<form action="mipagina.php" method="post">
  <p>Introduce tu nombre:</p>
  <input type="text" name="nombre" size="20" placeholder="Nombre completo">
  <br>

  <p>Introduce tu e-mail:</p>
  <input type="email" name="email" size="20" maxlength="50">
  <br>

  <input type="checkbox" name="casado"> Casado
  <p>Elige tu idioma preferido:</p>
  <input type="radio" name="idioma" value="ingles" checked>Inglés
  <br>
  <input type="radio" name="idioma" value="frances">Francés
  <br>

  <input type="submit" value="Enviar datos">
</form>
```

Esto se vería más o menos así en la página:

Introduce tu nombre:

Introduce tu e-mail:

Casado

Elige tu idioma preferido:

Inglés

Francés

Observa algunas cosas interesantes, como que los controles de tipo radio comparten el mismo name (para que al pulsar uno se suelten los demás), y que tienen un value que no se muestra en el formulario realmente, y cuya utilidad escapa al alcance de este tema.

2.1.1. Más sobre los botones

Además de poder definir botones con la etiqueta `input`, también podemos emplear la etiqueta `button` como alternativa. Tiene, igualmente, un atributo `type` que indica el tipo de botón que queremos crear

(*submit*, *reset* o *button* normal). Así, estas dos opciones son equivalentes (aunque, si observamos, la etiqueta *button* no es una etiqueta vacía, y el texto del botón se especifica en su contenido):

```
<input type="submit" value="Enviar datos">
<button type="submit">Enviar datos</button>
```

2.2. Las etiquetas de formularios: *label*

Con la etiqueta `label` podemos definir controles de tipo etiqueta, es decir, controles que sirven para dar información sobre otros controles. En realidad, en el ejemplo anterior, no deberíamos haber usado párrafos para indicar que el primer cuadro de texto es para escribir el e-mail, o que los botones de radio son para elegir el idioma preferido. Para eso están las etiquetas. La etiqueta *label* tiene un atributo `for`, que sirve para indicar el control al que está asociado (a través del atributo `id` de dicho control) para que, al hacer clic en la etiqueta, se acceda directamente al control. Además, el uso de etiquetas facilita la lectura de formularios por parte de lectores de pantalla, por lo que su uso es todavía más recomendable en términos de accesibilidad.

Así, el formulario anterior se podría haber definido así:

```
<form action="mipagina.php" method="post">
  <label for="nombre">Introduce tu nombre:</label>
  <br>
  <input type="text" name="nombre" id="nombre" size="20">
  <br>

  <label for="email">Introduce tu e-mail:</label>
  <br>
  <input type="email" name="email" id="email" size="20" maxlength="50">
  <br>

  <input type="checkbox" name="casado" id="casado">
  <label for="casado">Casado</label>
  <br>

  <label>Elige tu idioma preferido:</label>
  <br>
  <input type="radio" name="idioma" id="ingles" value="ingles" checked>
  <label for="ingles">Inglés</label>
  <br>
  <input type="radio" name="idioma" id="frances" value="frances">
  <label for="frances">Francés</label>
  <br>

  <input type="submit" value="Enviar datos">
</form>
```

Alternativamente, podemos hacer que la etiqueta englobe el control al que va a asociada. De este modo podemos omitir el atributo *for*, porque ya se sabe a qué control va asociada:

```
<label>
  Introduce tu nombre:
  <input type="text" name="nombre" id="nombre" size="20">
</label>
```

Otra de las ventajas que tiene usar etiquetas es que, en controles como los *checkboxes* o los botones de radio, ampliamos el área en que podemos hacer clic para seleccionarlos, ya que también podemos hacer clic en la etiqueta asociada al control (cosa que no se podría hacer si usáramos simples párrafos).

2.3. Áreas de texto

Para introducir texto que ocupe más de una línea, no nos sirve la etiqueta *input* con su atributo *type="text"*. En su lugar, debemos usar la pareja de etiquetas `textarea`. Esta etiqueta tiene tres atributos:

- `name` para ponerle un nombre al control, al igual que hacíamos con *input*

- `rows` para indicar cuántas filas de texto va a haber
- `cols` para indicar cuántas columnas de texto va a haber

Si luego se ocuparan más filas o columnas que las que indiquemos, aparecerá una barra de desplazamiento para poder ver el resto del texto escrito en el área de texto.

Aquí tenemos un ejemplo de uso:

```
<textarea name="noticia" rows="5" cols="20">
Texto de la noticia
</textarea>
```

2.4. Listas

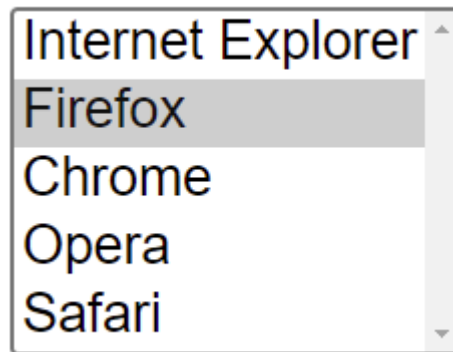
Podemos definir dos tipos de listas para elegir opciones en un formulario: listas fijas (ocupan una altura determinada y dejan visibles X elementos, aunque pueden tener más y verlos mediante barras de desplazamiento) y listas desplegables (sólo es visible un elemento, y los demás se muestran desplegando la lista). Ambas se crean con la etiqueta `select`, que admite los siguientes atributos:

- `name`: para darle un nombre al control, al igual que en controles anteriores
- `size`: para indicar el número de filas visibles. Si no usamos este atributo, o ponemos un valor de 1, crearemos una lista desplegable. Si vale más de 1, será una lista fija con el número de filas visibles que hayamos establecido aquí. En este caso, podemos utilizar un atributo llamado `multiple` (sin valor) que permite seleccionar varios elementos.

Internamente, la etiqueta `select` tiene una serie de etiquetas `option`, una para cada elemento que queramos añadir a la lista. Estos elementos tienen un atributo `value` para indicar su valor (que puede ser diferente del que luego se mostrará). Además, para indicar si hay algún elemento preseleccionado de antemano, marcaremos ese elemento con el atributo `selected`.

Veamos aquí un ejemplo y cómo se vería:

```
<select name="navegador" size="5">
  <option value="ie">Internet Explorer</option>
  <option value="firefox" selected>Firefox</option>
  <option value="chrome">Chrome</option>
  <option value="opera">Opera</option>
  <option value="safari">Safari</option>
</select>
```



Ejercicio 1:

Crema una página llamada **formDatos.html** para rellenar tus datos personales. Añade un encabezado 1 (*h1*) que diga "Rellena tus datos", y debajo un formulario como el que tienes a continuación:

Rellena tus datos

Nombre

Apellidos

E-mail

DNI

Idioma materno

- Castellano
- Valenciano
- Otros

Subir mi foto Ningún archivo seleccionado

Permitir que otros puedan ver mis datos


Ejercicio 2:

Crema una página llamada **formMatricula.html** para simular un proceso de matrícula en ciclos formativos:

Formulario de matrícula

Nombre completo:

Correo electrónico:

Fecha de nacimiento: 

Hermanos en el centro:

Teléfono móvil de contacto:

Ciclo seleccionado:

ASIR
ASIR
DAM
DAW

Deberás tener en cuenta que:

- El campo *Fecha de nacimiento* es de tipo *date*
- El campo *Correo electrónico* es de tipo *email*
- El campo *Hermanos en el centro* es de tipo numérico (*number*)

3. Algunas opciones avanzadas

3.1. Validación de formularios en cliente

Desde HTML5, podemos incorporar una serie de atributos en los controles de los formularios para establecer una validación básica. De este modo, podemos obligar a que algunos campos se rellenen, incluso que tengan un patrón o valor determinado. Para ello, podemos hacer uso de estos atributos:

required

Este atributo se puede aplicar desde HTML 5 en los campos de un formulario que sea obligatorio cumplimentar, de forma que el formulario directamente no se envía si están vacíos. Por ejemplo, en el siguiente formulario es obligatorio rellenar el campo login:

```
<form action="login.php" method="post">
  <label for="login">Login:</label>
  <input type="text" name="login" required>
  <br>
  <label for="login">Password:</label>
  <input type="text" name="password">
  <br>
  <input type="submit" value="Enviar">
</form>
```

Si no lo rellenamos, el navegador nos posicionará sobre ese cuadro de texto con algún tipo de mensaje (dependiendo del navegador) indicando que debemos rellenarlo.

pattern

Este atributo permite especificar un patrón de entrada, de forma que para que el dato que introduce el usuario se considere válido, debe ajustarse a ese patrón. Por ejemplo, para un campo que pida un número de teléfono podemos esperar que el usuario escriba entre 9 y 11 dígitos:

```
<input type="text" name="telefono" pattern="[0-9]{9,11}"
placeholder="Entre 9 y 11 dígitos">
```

Es habitual también indicar en ese campo del formulario qué tipo de información se espera a través de un *placeholder*. La sintaxis que se utiliza para definir diferentes tipos de patrones es la misma que se usa en JavaScript para expresiones regulares, por ejemplo, pero no la veremos con detalle en este tema. Simplemente veremos algunos casos útiles:

- Los **corchetes** especifican un **rango** de valores. Por ejemplo, `[0-9]` indica cualquier dígito, y `[a-zA-Z]` cualquier letra.
- Las **llaves** indican el **número de ocurrencias**, bien exacto (un solo número entre llaves) o un número mínimo y máximo de ocurrencias, separados por comas.
- El símbolo `+` indica una o más veces. Por ejemplo, uno o más dígitos (`[0-9]+`). Por su parte, el símbolo `*` indica cero o más veces.
- El símbolo `\` se utiliza para representar algunos símbolos especiales, como por ejemplo el punto (`\.`).

min, max y otras validaciones numéricas

Para campos numéricos o de tipo rango (*range*) podemos establecer con estos dos atributos los valores mínimo y/o máximo que se permiten (inclusive).

```
<input type="number" name="numero" min="1" max="5">
```

Además, podemos definir un atributo `step` para indicar el incremento que se admite. Esto permitirá definir si admitimos sólo números enteros, por ejemplo...

```
<input type="number" name="numero" min="1" max="5" step="1">
```

... o si también admitimos números reales, hasta las centésimas, por ejemplo:

```
<input type="number" name="numero" min="1" max="5" step="0.01">
```

Estos elementos también pueden aplicarse a los campos de tipo *range*:

```
<input type="range" name="porcentaje" min="0" max="100" step="5">
```

Otras validaciones *de facto*

Algunos controles de formularios incorporados desde HTML5, como el `input type="email"`, `input type="url"` o `input type="number"` incorporan una validación *de facto* en ellos, de forma que si el contenido que escribimos no se asemeja a un e-mail válido o a un dato numérico, respectivamente, también se mostrará un mensaje de error y no se enviará el formulario.

Ejercicio 3:

Vamos a aplicar validación al formulario del *Ejercicio 2*. En concreto, deberá cumplir lo siguiente:

- El nombre completo y el e-mail son obligatorios
- El móvil de contacto debe empezar por 6, 7 u 8, y contener en total 9 dígitos
- El número de hermanos admite valores entre 0 y 10, ambos inclusive

3.2. Conjuntos de campos o *fieldset*s

Podemos agrupar conjuntos de controles en una especie de "marcos", para poder dividir mejor la información del formulario cuando éste muestra demasiados campos. Estos marcos se crean con la pareja de etiquetas `fieldset`. Esta etiqueta contiene una subetiqueta `legend` que es el título de ese marco. Un ejemplo:

```
<fieldset>
  <legend>Datos personales</legend>
  <label for="nombre">Nombre</label>
  <br>
  <input type="text" name="nombre">
  <br>
  <label for="apellidos">Apellidos</label>
  <br>
  <input type="text" name="apellidos">
  <br>
  <label for="dni">DNI</label>
  <br>
  <input type="text" name="dni" size="10" maxlength="9">
  <br>
</fieldset>
```

que se vería así en el navegador:

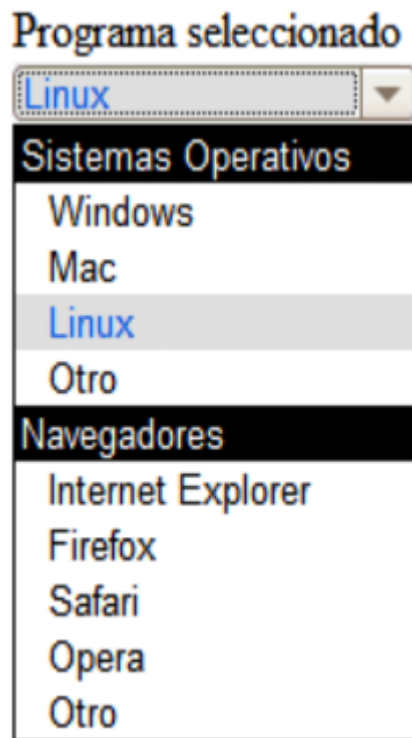


The image shows a browser rendering of the HTML code above. It features a rectangular box with a thin border. At the top left of the box is the title "Datos personales" in a bold, dark blue font. Below the title are three text input fields, each preceded by a label in a bold, dark blue font. The first field is labeled "Nombre", the second "Apellidos", and the third "DNI". Each input field is a simple white rectangle with a thin border, and they are arranged vertically with a small gap between them.

3.3. Agrupar opciones en listas

La etiqueta `select` admite algunas opciones más para agrupar los elementos que contiene en categorías, cuando hay muchos elementos que mostrar. Para ello se utiliza la etiqueta `optgroup`, con un atributo `label` que da nombre a cada categoría. Veamos un ejemplo y cómo se vería en el navegador:

```
<label for="programa">Programa seleccionado</label>
<br>
<select name="programa">
  <optgroup label="Sistemas Operativos">
    <option value="Windows">Windows</option>
    <option value="Mac">Mac</option>
    <option value="Linux" selected>Linux</option>
    <option value="Otro">Otro</option>
  </optgroup>
  <optgroup label="Navegadores">
    <option value="IE">Internet Explorer</option>
    <option value="Firefox">Firefox</option>
    ...
  </optgroup>
</select>
```



Ejercicio 4:

Crea una página llamada **formProducto.html** con el siguiente formulario (más el encabezado 1 con el texto "Información sobre el producto", y un botón de "Enviar" en la parte inferior):

Información sobre el producto

Datos básicos

Nombre

Descripción

Foto

Añadir contador de visitas

Datos económicos

Precio € Impuestos

Promoción

Ninguno

Transporte gratuito

Descuento 5%