Unidad 8

Formularios

”Para dialogar, preguntad primero;
después..., escuchad.”
Fray Antonio de Guevara

El formulario es un elemento indispensable para las aplicaciones web ya que es la forma más sencilla que tenemos de interactuar con nuestra aplicación web de forma que podamos enviar mas datos que los que podemos transmitir con un click de ratón.

El elemento que sirve para identificar un formulario lleva la etiqueta **form**. Dentro de él tendremos diferentes elementos por cada objeto que queramos mostrar en el mismo. Veamos el ejemplo el formulario más sencillo:

```
<form name="form2ASI" action="respuesta.html">
       <p><label for=”nombre”>Nombre:</label>
       <input type="text" id=”nombre” name="cajanombre" /></p>
       <p><input type="submit" value="Enviar" name="botonenv" /></p>
</form>
```

El elemento **form** tiene un único argumento imprescindible (**action**) que indica lo que hay que hacer para procesar el funcionario. En el ejemplo anterior se le pide que cargue la página **respuesta.html** que en realidad no hace nada y sólo muestra un mensaje. Lo normal sería llamar a un script cgi o una función javascript que validara los datos, a una página php que los procese en el lado del servidor o, mucho más rudimentario, enviar los resultados por correo electrónico con un mailto (por ejemplo **action="mailto:josemaria@uponaday.net"**). En este

último caso debería de aparecer también el argumento **method** y su valor debería de ser **post** (lo veremos a continuación).

El argumento **method** indica la forma en que el formulario envía los datos que el usuario ha escrito o seleccionado en él. Tiene dos valores posibles: **get** y **post.** Por defecto toma el valor **get**.

El método **get** envía los datos del formulario concatenados a la URL de la página. Los limita a 100 caracteres y es un método muy útil cuando, por ejemplo, queremos guardar el resultado en favoritos o enviárselo a alguien por correo. La URL que obtendríamos después de escribir Pepe y pulsar el botón de enviar en el ejemplo anterior debería de ser algo similar a esto:

```
http://10.0.0.1/ej2/respuesta.html?cajanombre=Pepe&botonenv=Enviar
```

El método **post** no tiene límite en su extensión y el resultado no es visible para el usuario pero, cuidado, eso no quiere decir que sea un método seguro para enviar información sensible. Ninguno de ambos métodos lo es puesto que el resultado de rellenar los datos del formulario se enviará en claro aún cuando con el método **post** no podamos verlos directamente.

En los siguientes puntos iremos viendo uno a uno los distintos elementos que pueden formar parte de un formulario y algunos de sus argumentos más usados. La mayor parte de ellos usan la etiqueta **input** e indican el tipo de objeto de que se trata mediante el valor del argumento **type**.

Caja de edición de texto (type=”text”)

Como su nombre indica, sirve para introducir texto con la restricción de que este no puede tener más de una línea.

```
<input type="text" name="nombre" size="20" maxlength="40" value=
"Escribe aquí tu nombre" />
```

El argumento **size** indica el tamaño en caracteres de la caja mientras que **maxlength** indica la longitud máxima que puede introducirse. **value** sirve para introducir un valor por defecto que aparecerá al cargar la página (en caso contrario aparecerá en blanco). **minlength** igualmente nos permitirá definir la longitud mínima válida

Cuando no queremos que el usuario edite el campo usaremos el parámetro **readonly** con cualquier valor (recuerda que en XHTML no está permitido la minimización de atributos o atributos sin valor, así que lo normal es poner **readonly="yes"** o **readonly="readonly"** aunque si ponemos **readonly="no"** o solamente **readonly** también deshabilitará la escritura en el campo). Además, pueden aparecer los argumentos **class** e **id** que nos serviran para aplicar estilos mediante CSS. El argumento **id** sirve, además, para asociar la caja de texto con una etiqueta (ver más adelante).

Con HTML5 se añaden nuevos argumentos. Son los siguientes:

El argumento **required**, válido para casi todos los tipos de objetos permite, marcar como obligatorio un campo de forma que no se pueda ejecutar la función submit si no está completo y dando, además, un efecto visual que cada navegador podrá personalizar.

El argumento **autofocus**, que debería de usarse sólo en un objeto por formulario, selecciona el campo del mismo que tomará el foco de forma automática una vez cargada la página.

El argumento **autocomplete** puede tomar dos valores (**on** o **off**) e indica al navegador si puede hacer sugerencias al usuario dependiendo de lo que haya escrito en anteriores ocasiones en ese mismo campo.

El argumento **placeholder**, válido en todos los campos de texto, nos muestra un texto con instrucciones sobre que debemos de cumplimentar en el mismo.

El argumento **wrap**, existente en versiones anteriores a HTML4 pero desaparecido en esta, vuelve en HTML5. Es válido en los input de tipo textarea y nos permite indicar si queremos que los saltos de línea que el usuario escriba en ellos se tranmsitan (**wrap="hard"**) o no (**wrap="soft"**) que es la opción por defecto.

datalist es una nueva etiqueta que puede asociarce a los campos de texto y que nos permite mostrar al usuario una lista de sugerencias de forma muy similar a la que, por ejemplo, Google utiliza cuando escribimos en su caja de búsquedas. Se usa en conjunto con la etiqueta option de forma muy parecida a la que utilizamos en un campo select:

```
<input type="text" name="provincia" list="provincias" />
<datalist id="provincias">
        <option value="Sevilla"></option>
        <option value="Madrid"></option>
        <option value="Cuenca"></option>
        <option value="Málaga"></option>
</datalist>
```

La identificación entre el input y el datalist se hace a través de los argumentos **list** e **id** cuyos valores deben de ser iguales.

Por último, el argumento **pattern** permite especificar una expresión regular que debe de cumplir lo que pongamos en el campo para que el formulario sea validado. Veamos un par de ejemplos:

```
<input type="text" name="codpostal" pattern="[0-9]{5}" />
<input type="text" name="nif" pattern="[0-9]{8}[A-Za-zÑñ]
{1}" />
```

En el primero el patrón especifica que el campo sólo validará si consta de cinco números seguidos sin ningún caracter adicional. En el segundo caso, el campo debe de ser rellenado con ocho números seguidos de una letra, ya sea esta en mayúsculas o minúsculas. Para aprender más sobre como construir expresiones regulares más complejas puedes consultar los siguientes enlaces:

https://support.google.com/a/answer/1371415?hl=es

https://4geeks.com/es/lesson/regex-tutorial-regular-expression-ejemplo

O buscar en Google que es tu amigo ;-)

Y una nota adicional acerca de los patrones: si el campo no está marcado como requerido mediante el argumento antes visto sería posible dejarlo en blanco y eso no cumpliría con el patrón especificado. Así que si se quiere obligar no olvides también este argumento.

> ***ATENCIÓN.*** *Muchos de estos argumentos que hemos visto centrados en los input de tipo text pueden usarse también con la mayoría del resto de las etiquetas que vamos a ver a continuación. Experimenta con ellos*

Caja de edición de contraseña (type="password")

Similar a la anterior pero en este caso lo que escribamos sobre ella aparecerá camuflado mediante puntos, asteriscos o algo similar (según el navegador). Los argumentos que podemos usar también son los mismos que en el anterior.

```
<input type="password" name="pwd" size="12" maxlength="12" />
```

Etiquetas (label)

El elemento label sirve para asociar un campo de texto con un objeto de edición de forma que, por ejemplo, si el usuario pulsa con el ratón sobre la etiqueta asociada a un campo de edición de texto el cursos aparecería en modo edición en el interior de dicho campo como si el usuario hubiera pulsado en él. Se trata, por tanto, de una forma de extender la superficie de dichos objetos de forma que resulte más cómodo la cumplimentación de los formularios.

Tenemos dos formas diferentes de realizar esta asociación. La primera de ellas consiste simplemente en anidar el elemento de edición dentro del contenido del elemento label:

```
<label>Dirección:<br/><input type="text" name="direccion" size="80"
maxlength="120" /></label>
```

El segundo método consiste en identificar ambos objetos mediante el valor del argumento **for** del elemento **label** y el valor del argumento **id** del elemento de edición:

```
<label for="txtdir">Dirección:</label><br/><input type="text"
name="direccion" size="80" maxlength="120" id="txtdir" />
```

> ***ATENCIÓN.*** *Si usamos el primero de los métodos hay que tener mucho cuidado porque los estilos CSS que apliquemos al elemento* ***label*** *se extenderían también al objeto de edición.*

Radio Buttons (type="radio")

Los radio buttons son los característicos objetos redondos para elegir entre varias opciones alternativas pero con la restricción de que sólo puede seleccionarse uno de ellos, de forma que marcar cualquiera implica que se elimina la selección del que hubiéramos seleccionado previamente. Además, debería de haber una elección obligatoria.

```
<input type="radio" name="sexo" value="hombre" checked="checked" />
Hombre<br/>
<input type="radio" name="sexo" value="mujer" /> Mujer
```

Puesto que en un mismo formulario podemos tener diferentes grupos de radio buttons, es el valor del argumento **name** el que los mantiene asociados entre si. El argumento **checked** sirve para seleccionar la opción por defecto que aparecerá marcada al cargar la página. Además, podemos usar el argumento **disabled** si queremos que alguno de ellos no sea seleccionable.

ATENCIÓN. *Cuidado con las propiedades que aparentemente funcionan pero no lo hacen. Si ponemos un argumento* ***readonly*** *en un radio (o en un check box que veremos a continuación), este cambia de aspecto pero puede seguir usándose. El argumento correcto para deshabilitarlos sería* ***disabled.***

El valor del argumento **value** es lo que se transmitirá en las respuesta del formulario si ese elemento se encuentra seleccionado.

Checkboxes (type="checkbox")

Los checkboxes son muy similares a los radio butons. Se diferencian básicamente en dos aspectos: visualmente son cuadrados y desde el punto de vista lógico deberían de permitir, si así lo deseamos, la elección de más de una opción dentro de un mismo grupo de objetos. Además, debería de permitirnos no elegir ninguno de ellos.

```
<input type="checkbox" name="vehiculo" value="motocicleta"
disabled="disabled" /> Motocicleta<br/>
<input type="checkbox" name="vehiculo" value="moto"
checked="checked" /> Moto<br/>
<input type="checkbox" name="vehiculo" value="coche" /> Coche<br/>
<input type="checkbox" name="vehiculo" value="camión" /> Camión<br/>
<input type="checkbox" name="vehiculo" value="tractor" /> Tractor</p>
```

Todo lo dicho en cuanto a los argumentos en los radio es válido también para estos.

Agrupación de objetos

Podemos agrupar visualmente diferentes objetos de nuestro formulario mediante el elemento **fieldset**:

```
<fieldset>
<legend>Información personal</legend>
...
</fieldset>
```

Esto, por defecto, dibuja un recuadro alrededor de los elementos que contiene. El elemento **legend** sirve para poner título a dicho recuadro.

Caja de edición de texto de múltiples líneas

El elemento **textarea** nos crea una caja de edición que admite más de una línea. Las dimensiones iniciales de la misma se asignan mediante los argumentos **rows** y **cols**.

```
<textarea name="comentarios" rows="8" cols="50" >Escribe tus
comentarios...</textarea>
```

textarea admite la mayoría de los atributos habituales de input de tipo text: placeholder, maxlength, disabled, readonly...

Todos los navegadores modernos permiten cambiar el tamaño de estas cajas de texto visualmente mediante un control que aparece en su esquina inferior derecha.

Caja de selección de ficheros

Nos abre una ventana de diálogo (cuyo aspecto depende del navegador y el sistema operativo) que nos permite seleccionar un fichero de nuestro disco duro

```
<label>Fotografía:<br/><input type="file" name="fichero"/></label>
```

No realiza ningún tipo de validación de la elección. Eso nos corresponde hacerlo a nosotros posteriormente mediante javascript o en el servidor cuando enviemos los datos.

Además de algunos de los ya vistos, tenemos atributos específicos para usar con este input:

accept nos permite filtrar el tipo de ficheros válidos

multiple nos permite selecionar mas de un fichero

```
<input type = “file” name=”documentos” multiple=”on”
accept=".doc,.docx,.odt, .pdf, .txt” />
```

El atributo capture nos permite usar la cámara y/o micrófono de nuestro equipo (si existe y está permitido su uso) para hacer una captura de fotos, vídeos o audio:

```
<label for="soundFile">Graba tu voz: <input type="file"
id="soundFile" capture="user" accept="audio/*"
/></label>

<label for="videoFile">Graba un vídeo: <input
type="file" id="videoFile" capture="environment"
accept="video/*" /></label>

<label for="imageFile">Toma una foto: <input type="file"
id="imageFile" capture="user" accept="image/*"
/></label>
```

Cajas de selección (Comboboxes y Listboxes)

Existen dos cajas de selección típicas en entornos gráficos: los comboboxes son cajas con distintas opciones en las que sólo podemos seleccionar una de ellas. Además, aunque podemos desplegarla en el momento de la selección, cuando elegimos la lista se cierra y sólo permanece visible nuestra elección. Además, debería de ser obligatorio hacer al menos una elección. Los listboxes, por el contrario, nos suelen permitir elegir más de una opción simultaneamente (o ninguna) y siempre suelen estar total o parcialmente abiertos.

En HTML ambos se implementa mediante el elemento **select** y son los argumentos que usemos los que hacen que se comporte de una u otra forma. Veamos inicialmente un combobox:

```
<select name="oficina" >
       <option value="madrid">Madrid</option>
       <option value="sevilla">Sevilla</option>
       <option selected="selected" value="alcorcon">Alcorcón</option>
       <option value="lisboa">Lisboa</option>
       <option value="burdeos">Burdeos</option>
</select>
```

Un listbox sería de esta forma:

```
<select name="aficiones" multiple="multiple" size="4" >
	<option value=”apatia”>Nada</option>
	<option value=”baile”>Bailar</option>
	<option value=”futbol”>Fútbol</option>
	<option value=”rol”>Juegos de Rol</option>
	<option value=”cocacola”>Ponerse tibio de beber
cocacola</option>
	<option value=”tumbing”>Dormir en el sofá</option>
</select>
```

Como vemos, la diferencia la marcan dos argumentos: **size** que indica cuantos elementos de la lista serán visibles simultaneamente (por defecto es sólo 1, que es como debería de comportarse un combobox) y **multiple** que es el argumento que indica que podemos hacer más de una selección. También podemos usar el argumento **disabled** ya visto en otros controles anteriores.

En las opciones se usa el argumento **selected** para indicar si queremos que alguna de las opciones aparezca inicialmente seleccionada.

Tanto en uno como en otro podemos usar el elemento **optgroup** para agrupar las distintas opciones en categorías:

```
<select name="oficina">
	<optgroup label="En España">
		<option value=”madrid”>Madrid</option>
		<option value=”sevilla”>Sevilla</option>
		<option value=”alcorcon” selected="selected"
>Alcorcón</option>
	</optgroup>
	<optgroup label="En el extranjero">
		<option value=”lisboa”>Lisboa</option>
		<option value=”burdeos”>Burdeos</option>
	</optgroup>
</select>
```

Botones

Todo formulario debería de tener, al menos, un botón que sirva para indicar que ya hemos terminado de cumplimentarlo y que da paso al procesado y/o envío de los datos que hemos rellenado. Este botón debe

de estar identificado con el argumento **type** y el valor **submit** y el comportamiento normal al pulsarlo es invocar el enlace que hemos puesto como valor del argumento **action** del elemento **form**. Dicho esto, tenemos dos formas de implementar este botón:

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

Ambos elementos se comportarían igual y harían lo mismo. La diferencia está en que el texto que aparece en el primero es el que aparece como valor del argumento **value** (o un texto por defecto que pone el navegador si ese argumento no aparece) y que, por tanto, no podemos aplicarle estilos mientras que en el segundo botón el texto es el contenido del elemento y podemos aplicar estilos con total libertad o, incluso, poner una imagen:

```
<button type="submit" ><img src="ok.jpg" alt="Ok" /></button>
```

Ambos tipos de botones admiten el argumento **disabled**.

El segundo tipo de botones "característicos" de un formulario es uno que nos permite limpiarlo y dejar todos los valores del mismo como si la página se hubiera cargada por primera vez. Se identifica mediante el argumento type con su valor a reset y podemos, igualmente, implementarlo de ambas formas. Todo lo dicho antes sirve también en este caso:

```
<input type="reset" />
<button type="reset" ><img src="clean.jpg" alt="Limpiar" /></button>
```

Tenemos, además, la posibilidad de introducir en nuestro formulario botones genéricos sin una acción previamente definida que posteriormente nosotros le asignaremos usando código javascript:

```
<input type="image" src="button.png" />
<input type="button" value="Botón" /><br/>
<button type="button">Botón</button>
```

El primero no es propiamente un botón, sino una imagen pero a la que también se le asocia por defecto las mismas propiedades que a un botón con el argumento **type** con valor de **submit**. Los otros dos son botones sin ningún tipo de acción asociada a los que posteriormente se les puede aplicar alguna usando código javascript.

Mejoras en los formularios: nuevos tipos de Input

Tal vez una de las mejoras más significativas sea la gran riqueza que HTML5 añade a los formularios con objeto de que podamos construir con facilidad aplicaciones web. Son 12 los nuevos tipos disponibles para la etiqueta **input**

Los tipos nuevos introducidos a los input son **email, url, number, date, month, week, time, datetime-local, search, color, range** y **tel.** Proporcionan controles especializados de diferentes peticiones bastante cotidianas con validación automática. Veamos algunos ejemplos con los argumentos más habituales de cada uno de ellos (aparte de los ya vistos: **required, disabled, readonly**, etc. y que funcionarían perfectamente con la mayoría de ellos).

```
<input type="email" name="email" />

<input type="url" name="web" />

<input type="number" name="edad" min="18" max="100" step="1"
value="18" />

<input name="cursor" name="unidades" type="range" min="3"
max="12" value="6" step="3" />

<input type="date" name="nacimiento"  />

<input type="month" name="alta" />

<input type="week" name="vacaciones" />
```

```
<input type="time" min="07:00" max="23:00" step="3600"
value="09:00" />

<input type="datetime-local" name="fecha" />

<input type="color" name="color" value="#c0ffee" />

<input type="search" name="busqueda" /> <input type="button"
value="Buscar" />

<input type="tel" name="telefono" />
```

> ***ATENCIÓN.*** *Puede que algunos de estos input no estén aún implementados en tu navegador. Prueba el código que aparece aquí arriba y si no funciona es por eso. También puedes comprobarlo en las W3Schools.*

El input tipo tel no realiza ninguna validación y su existencia obedece sólo a motivos semánticos. Si queremos realizar una validación tenemos que hacerla nosotros mismos usando los atributos vistos al principio de este capítulo

Intimamente relacionado con los formularios, aunque mucho más universal y sin dependencia directa con estos, tenemos el nuevo agumento **contenteditable** que convierte, cuando su valor es **true**, a casi cualquier elemento, ya sea de bloque o de linea, en editable de forma directa por el usuario:

```
<h1 contenteditable="true">Titular editable</h1>
<p contenteditable="true">Este párrafo también es editable</p>
<div contenteditable="true">Y este bloque div lo mismo</div>
<p>Pero, incluso <strong contenteditable="true">estas
negritas</strong>  o este <span contenteditable="true">texto
con un span</span> que son elementos de línea pueden ser
editables</p>
```