

Imágenes, formatos gráficos e optimización

Vemos cómo colocar una imagen en una página web y algunos atributos básicos para asignarle estilos a las imágenes en HTML.

Sin duda uno de los aspectos más vistosos y atractivos de las páginas web es el grafismo. La introducción en nuestro texto de imágenes puede ayudarnos a explicar más fácilmente nuestra información y darle un aire mucho más estético. El abuso no obstante puede conducirnos a una sobrecarga que se traduce en una distracción para el navegante, quien tendrá más dificultad en encontrar la información necesaria.

El uso de imágenes también tiene que ser realizado con cuidado porque aumentan el tiempo de carga de la página, lo que puede ser de un efecto nefasto si nuestro visitante no tiene una buena conexión o si es un poco impaciente. Por ello es recomendable siempre optimizar las imágenes para Internet, haciendo que su tamaño en bytes sea lo mínimo posible, para facilitar la descarga, pero sin que ello comprometa mucho su calidad.

Las imágenes son almacenadas en forma de archivos, principalmente GIF (para dibujos) o JPG (para fotos). Estos archivos los podemos obtener desde diversas vías, como por ejemplo nuestra cámara digital, aunque también pueden ser creados por nosotros mismos con algún editor gráfico o pueden ser descargados gratuitamente en sitios web especializados.

La etiqueta que utilizaremos para insertar una imagen es IMG (image). Esta etiqueta no posee su cierre correspondiente y en ella hemos de especificar obligatoriamente el paradero de nuestro archivo gráfico mediante el atributo src (source).

La sintaxis queda entonces de la siguiente forma:

```

```

Aparte de este atributo, indispensable obviamente para la visualización de la imagen, la etiqueta IMG nos propone otra serie de atributos de mayor o menor utilidad, que listamos a continuación:

Atributos height y width

Estos atributos definen la altura y anchura respectivamente de la imagen en píxeles. Aunque estas dimensiones forman parte del estilo de la imagen, y por tanto podrían ir en el CSS, todavía puede ser interesante definir las dentro del HTML. De nuevo, en 2016 ya no es tan indispensable, puesto que muchos sitios creados con "[Responsive Web Design](#)" prefieren que las imágenes se adapten al tamaño de la pantalla donde se va a visualizar.

Todos los archivos gráficos poseen unas dimensiones de ancho y alto. Estas dimensiones pueden obtenerse a partir del propio diseñador gráfico o bien haciendo clic con el botón derecho sobre la imagen, vista desde el explorador de archivos de tu ordenador, para luego elegir "propiedades" o "información de la imagen" sobre el menú que se despliega. Un ejemplo de etiqueta IMG con sus valores de anchura y altura declarados te quedaría así:

```

```

Imágenes que son enlaces

Ni que decir tiene que una imagen, lo mismo que un texto, puede servir de enlace. Vista la estructura de los [enlaces en HTML](#), podemos muy fácilmente adivinar el tipo de código necesario:

```
<a href="archivo.html"></a>
```

Formatos gráficos para páginas web

Presenta los formatos gráficos utilizados en las páginas web, el GIF, el JPG y PNG. Hace hincapié en los dos primeros, resumiendo sus características y enseñando a optimizar los ficheros.

Tipos de archivos

En Internet se utilizan principalmente dos tipos de archivos gráficos GIF y JPG, pensados especialmente para optimizar el tamaño que ocupan en disco, ya que los archivos pequeños se transmiten más rápidamente por la Red.

El formato de archivo GIF se usa para las imágenes que tengan dibujos, mientras que el formato JPG se usa para las fotografías. Los dos comprimen las imágenes para guardarlas. La forma de comprimir la imagen que utiliza cada formato es lo que los hace ideales para unos u otros propósitos.

Adicionalmente, se puede usar un tercer formato gráfico en las páginas web, el PNG. Este formato no tiene tanta aceptación como el GIF o JPG por varias razones, entre las que destacan el desconocimiento del formato por parte de los desarrolladores, que las herramientas habituales para tratar gráficos (como por ejemplo Photoshop) generalmente no lo soportaban y que los navegadores antiguos también tienen problemas para visualizarlas.

Tablas en HTML

Las tablas fueron muy importantes en una época para maquetar páginas web. Hoy lo adecuado es utilizarlas sólo para presentar información tabulada, es decir, colocada en una rejilla de filas y columnas. En los siguientes artículos aprenderemos todo sobre las tablas en HTML.

Tablas en HTML

Vemos lo que son las tablas, para que sirven y en qué casos podemos utilizarlas. Vemos la tabla más simple posible. Una tabla en un conjunto de celdas organizadas dentro de las cuales podemos alojar distintos contenidos.

HTML dispone de una gran variedad de etiquetas para crear tablas, con sus atributos, de las cuales veremos una introducción en este artículo. En un principio nos podría parecer que las tablas son raramente útiles y que pueden ser utilizadas principalmente para listar datos como agendas, resultados y otros datos de una forma organizada. En general, sirven para representar información tabulada, en filas y columnas. Esto es una realidad en los últimos años, desde que las tablas se han descartado para fines relacionados con la maquetación.

Etiquetas básicas para tablas en HTML

Para empezar, nada más sencillo que por el principio: las tablas son definidas por las etiquetas TABLE y su cierre.

Dentro de estas dos etiquetas colocaremos todas las otras etiquetas de las tablas, hasta llegar a las celdas. Dentro de las celdas ya es permitido colocar textos e imágenes que darán el contenido a la tabla.

Las tablas son descritas por líneas de arriba a abajo (y luego por columnas de izquierda a derecha). Cada una de estas líneas, llamada fila, es definida por otra etiqueta y su cierre: TR

Asimismo, dentro de cada línea, habrá diferentes celdas. Cada una de estas celdas será definida por otra etiqueta: TD. Dentro de ésta y su cierre será donde coloquemos nuestro contenido, el contenido de cada celda. Aquí tenéis un ejemplo de estructura de tabla:

```
<table>
<tr>
  <td>Celda 1, línea 1</td>
  <td>Celda 2, línea 1</td>
</tr>
<tr>
  <td>Celda 1, línea 2</td>  <td>Celda 2, línea 2</td>
</tr>
</table>
```

El resultado:

Celda 1, línea 1	Celda 2, línea 1
Celda 1, línea 2	Celda 2, línea 2

Atributos para tablas, filas y celdas

A partir de esta idea simple y sencilla, las tablas adquieren otra magnitud cuando les incorporamos toda una batería de atributos aplicados sobre cada tipo de etiquetas que las componen.

En cuanto a atributos para tabla hay unos cuantos. Muchos los conoces ya de otras etiquetas, como width, height, align, etc. Hay otros que son especialmente creados para las etiquetas TABLE.

cellspacing: es el espacio entre celdas de la tabla.

cellpadding: es el espacio entre el borde de la celda y su contenido.

border: es el número de píxeles que tendrá el borde de la tabla.

bordercolor: es el rgb que le vas a asignar al borde de la tabla.

Así pues, podemos especificar el formato de nuestras celdas a partir de etiquetas introducidas en su interior o mediante atributos colocados dentro de la etiqueta de celda TD o bien, en algunos casos, dentro de la etiqueta TR, si deseamos que el atributo sea válido para toda la línea. La

forma más útil y actual de dar forma a las celdas es a partir de las [hojas de estilo en cascada](#).

Veamos a continuación algunos atributos útiles para la construcción de nuestras tablas. Empecemos viendo atributos que nos permiten modificar una celda en concreto o toda una línea:

align: Justifica el texto de la celda del mismo modo que si fuese el de un párrafo.

valign: Podemos elegir si queremos que el texto aparezca arriba (top), en el centro (middle) o abajo (bottom) de la celda.

bgcolor: Da color a la celda o línea elegida.

bordercolor: Define el color del borde.

Otros atributos que pueden ser únicamente asignados a una celda y no al conjunto de celdas de una línea son:

background: Nos permite colocar un fondo para la celda a partir de un enlace a una imagen.

height: Define la altura de la celda en pixels o porcentaje.

width: Define la anchura de la celda en pixels o porcentaje.

colspan: Expande una celda horizontalmente.

rowspan: Expande una celda verticalmente.

A título de ejemplo:

```
<td width="80">
```

Dará una anchura de 80 pixels a la celda. Sin embargo,

```
<td width="80%">
```

Dará una anchura a la celda del 80% de la anchura de la tabla. Hay que tener en cuenta que, definidas las dimensiones de las celdas, el navegador va a hacer lo que buenamente pueda para satisfacer al programador. Esto quiere decir que puede que en algunas ocasiones el resultado que obtengamos no sea el esperado. Concretamente, si el texto presenta una palabra excesivamente larga, puede que la anchura de la celda se vea aumentada para mantener la palabra en la misma línea. Por otra parte, si el texto resulta muy largo, la celda aumentará su altura para poder mostrar todo su contenido.

Análogamente, si por ejemplo definimos dos anchuras distintas a celdas de una misma columna, el navegador no sabrá a cuál hacer caso. Es por ello que resulta conveniente tener bien claro desde un principio como es la tabla que queremos diseñar. No está de más si la prediseñamos en papel si la complejidad es importante. El HTML resulta en general fácil

pero las tablas pueden convertirse en un verdadero quebradero de cabeza si no llegamos a comprenderlas debidamente.

Los atributos `rowspan` y `colspan` son también utilizados frecuentemente. Gracias a ellos es posible expandir celdas fusionando éstas con sus vecinas. El valor que pueden tomar estas etiquetas es numérico. El número representa la cantidad de celdas fusionadas. Así:

```
<td colspan="2">
```

Fusionara la celda en cuestión con su vecina derecha.

Esta celda tiene un `colspan="2"`

Celda normal

Otra celda

Del mismo modo,

```
<td rowspan="2">
```

Expandirá la celda hacia abajo fusionándose con la celda inferior.

Esta celda tiene `rowspan="2"`,
por eso tiene fusionada la
celda de abajo.

Celda
Normal
Otra
celda
normal

Tablas anidadas

Muy útil también es el uso de tablas anidadas. De la misma forma que podíamos incluir listas dentro de otras listas, las tablas pueden ser incluidas dentro de otras. Así, podemos incluir una tabla dentro de la celda de otra. El modo de funcionamiento sigue siendo el mismo, aunque

la situación puede complicarse si el número de tablas embebidas dentro de otras es elevado.

Vamos a ver un código de anidación de tablas. Veamos primero el resultado y luego el código, así conseguiremos entenderlo mejor.

Celda de la tabla principal	Tabla anidada, celda 1	Tabla anidada, celda 2
	Tabla anidada, celda 3	Tabla anidada, celda 4

Este sería el código:

```
<table cellpadding="10" cellspacing="10"
border="3">
<tr>
<td align="center">
Celda de la tabla principal
</td>
<td align="center">
<table cellpadding="2" cellspacing="2" border="1">
<tr>
<td>Tabla anidada, celda 1</td>
<td>Tabla anidada, celda 2</td>
</tr>
<tr>
<td>Tabla anidada, celda 3</td>
<td>Tabla anidada, celda 4</td>
</tr>
</table>
</td>
</tr>
</table>
```

Ejemplos prácticos

Hasta aquí la información que pretendíamos transmitir sobre las tablas en HTML. Sería importante ahora realizar algún ejemplo de realización de una tabla un poco compleja. Por ejemplo la siguiente:

Animales en peligro de extinción			
Nombre	Cabezas	Previsión 2010	Previsión 2020
Ballena	6000	4000	1500
Oso Pardo	50	0	
Lince	10		
Tigre	300	210	

Se puede ver el código fuente para generar esa tabla. Pero antes intenta realizarla por ti mismo, que es esencial para poder afianzar el conocimiento. Ten en cuenta también que ciertos estilos colocados en la tabla pueden no funcionar en todos los navegadores. Además, lo que hemos repetido ya innumerables veces: los estilos forman parte de la responsabilidad del [CSS](#).

```
<table align="center" cellspacing="2"
cellpadding="2" border="1" bgcolor=dddddd>
<tr>
<td colspan="4" align="center"
bgcolor="666666"><font
color="#FFFFFF"><strong>Animales en peligro de
extinción</strong></font></td>
</tr>
<tr bgcolor="aaaaaa">
<td>Nombre</td>
<td align="center">Cabezas</td>
<td align="center">Previsión 2010</td>
<td align="center">Previsión 2020</td>
</tr>
<tr>
<td>Ballena</td>
<td align="center">6000</td>
<td align="center">4000</td>
<td align="center">1500</td>
</tr>
<tr>
<td>Oso Pardo</td>
<td align="center">50</td>
```



```

<td rowspan="2" colspan="2" align="center"
bgcolor="red">0</td>
</tr>
<tr>
<td>Lince</td>
<td align="center">10</td>
</tr>
<tr>
<td>Tigre</td>
<td align="center">300</td>
<td colspan="2" align="center">210</td>
</tr>
</table>

```

Otro ejemplo de tabla con el que podemos practicar. En este caso hemos implementado una anidación de tablas, es decir, dentro de un TD hemos colocado un TABLE completo. Es un buen ejemplo para seguir aprendiendo.

Climas de América del Sur			
Parte de arriba de América del Sur. Países como:	Venezuela	Parte de abajo de América del Sur. Países como:	Argentina
	Colombia		Chile
	Ecuador		Uruguay
	Perú		Paraguay
Bosque tropical, clima de sabana, clima marítimo con inviernos secos.		Climas marítimos con veranos secos, con inviernos secos, climas fríos, clima de estepa, clima desértico.	

También podemos ver su código fuente.

```

<table cellpadding="4" cellspacing="4" border="1"
width=400 bgcolor=dddddd>
<tr>
<td colspan="2" bgcolor="666666"
align="center"><font

```

```

color="#FFFFFF"><strong>Climas de América del
Sur</strong></font></td>
</tr>
<tr>
<td width="50%">
<table align="right" cellspacing="1"
cellpadding="1" border="1">
<tr>
<td bgcolor="#cccccc"
align="center">Venezuela</td>
</tr>
<tr>
<td bgcolor="#cccccc" align="center">Colombia</td>
</tr>
<tr>
<td bgcolor="#cccccc" align="center">Ecuador</td>
</tr>
<tr>
<td bgcolor="#cccccc" align="center">Perú</td>
</tr>
</table>
Parte de arriba de América del Sur. Países como:
</td>
<td width="50%">
<table align="right" cellspacing="1"
cellpadding="1" border="1">
<tr>
<td bgcolor="#cccccc"
align="center">Argentina</td>
</tr>
<tr>
<td bgcolor="#cccccc" align="center">Chile</td>
</tr>
<tr>
<td bgcolor="#cccccc" align="center">Uruguay</td>
</tr>
<tr>
<td bgcolor="#cccccc" align="center">Paraguay</td>
</tr>
</table>
Parte de abajo de América del Sur. Países como:
</td>
</tr>
<tr>

```

```
<td bgcolor="#358391">Bosque tropical, clima de  
sabana, clima marítimo con inviernos secos.</td>  
<td bgcolor="#358391">Climas marítimos con veranos  
secos, con inviernos secos, climas frios, clima de  
estepa, clima desértico.</td>  
</tr>  
</table>
```

Formularios en HTML

El trabajo con formularios es uno de los principales puntos que debemos aprender en HTML. Hacen posible muchas de las utilidades clave de una web, como el contacto de los creadores de las páginas con los visitantes, así como ciertos niveles de interacción básica y avanzada con el usuario.

Formularios HTML

Empezamos la explicación de la creación de formularios con el lenguaje HTML.

Hasta ahora hemos visto la forma en la que el HTML gestiona y muestra la información, esencialmente mediante texto, imágenes y enlaces. Nos queda por ver de qué forma podemos intercambiar información con nuestro visitante. Desde luego, este nuevo aspecto resulta primordial para gran cantidad de acciones que se pueden llevar a cabo mediante el Web: Comprar un artículo, rellenar una encuesta, enviar un comentario al autor...

Qué se puede hacer con un formulario

Usando HTML podemos únicamente enviar el contenido del formulario a un correo electrónico, es decir, construir un formulario con diversos campos y, a la hora pulsar el botón de enviar, generar una ventana de redacción de un email con los datos que el usuario haya escrito en cada uno de esos campos.

A menudo desearemos hacer cosas más complejas con los formularios, como que se envíe automáticamente el correo a un email sin necesidad que el contenido pase por ningún programa de email. Para ello tendremos que procesar el formulario mediante un programa.

La cosa puede resultar un poco más compleja, ya que tendremos que emplear otros lenguajes más sofisticados que el propio HTML. En este caso, la solución más sencilla es utilizar los programas prediseñados que nos ofrecen un gran número de servidores de alojamiento y que nos

permiten almacenar y procesar los datos en forma de archivos u otros formatos. Si vuestras páginas están alojadas en un servidor que no os propone este tipo de ventajas, siempre podéis recurrir a servidores de terceros que ofrecen este u otro tipo de [servicios gratuitos para webs](#). Por supuesto, existe otra alternativa que es la de aprender lenguajes como [ASP](#) o [PHP](#) que nos permitirán, entre otras cosas, el [tratamiento de formularios](#).

Así pues, en resumen, con HTML podremos construir los formularios, con diversos tipos de campos, como cajas de texto, botones de radio, cajas de selección, menús desplegables, etc. Sin embargo, debe quedar claro que desde HTML no se puede enviar directamente el correo, sino que se generará un email en el ordenador del visitante, que éste tendrá que enviar "manualmente" por medio de su programa de correo. Si queremos que el formulario se envíe automáticamente o se procese en el servidor para generar otro tipo de respuesta, necesitaremos lenguajes de programación.

Cómo hacer un formulario en HTML

Los formularios son definidos por medio de las etiquetas FORM y su cierre. Entre estas dos etiquetas colocaremos todos los campos y botones que componen el formulario. Dentro de esta etiqueta FORM debemos especificar algunos atributos:

action: define el tipo de acción a llevar a cabo con el formulario. Como ya hemos dicho, existen dos posibilidades:

- El formulario es enviado a una dirección de correo electrónico
- El formulario es enviado a un programa o script que procesa su contenido

En el primer caso, el contenido del formulario es enviado a la dirección de correo electrónico especificada por medio de una sintaxis de este tipo:

```
<form action="mailto:direccion@correo.com" ...>
```

Si lo que queremos es que el formulario sea procesado por un programa, hemos de especificar la dirección del archivo que contiene dicho programa. La etiqueta quedaría en este caso de la siguiente forma:

```
<form action="dirección del archivo" ...>
```

La forma en la que se expresa la localización del archivo que contiene el programa es la misma que la [vista para los enlaces](#).

method: Este atributo se encarga de especificar la forma en la que el formulario es enviado.

Los dos valores posibles que puede tomar esta atributo son post y get. A efectos prácticos y, salvo que se os diga lo contrario, daremos siempre el valor post.

enctype: Se utiliza para indicar la forma en la que viajará la información que se mande por el formulario. En el caso más corriente, enviar el formulario por correo electrónico, el valor de este atributo debe de ser "text/plain". Así conseguimos que se envíe el contenido del formulario como texto plano dentro del email.

Si queremos que el formulario se procese automáticamente por un programa, generalmente no utilizaremos este atributo, de modo que tome su valor por defecto, es decir, no incluiremos enctype dentro de la etiqueta FORM.

Ejemplo de etiqueta FORM completa

Así, para el caso más habitual -el envío del formulario por correo- la etiqueta de creación del formulario tendrá el siguiente aspecto:

```
<form action="mailto:direccion@correo.com (o  
nombre del archivo de proceso)" method="post"  
enctype="text/plain">
```

Entre esta etiqueta y su cierre colocaremos el resto de etiquetas que darán forma a nuestro formulario, las cuales serán vistas en capítulos siguientes.

Para continuar, vamos a ver cómo insertar cada uno de los campos posibles en un formulario HTML, comenzando por los [campos de texto](#).

Elementos de Formularios. Campos de texto

Vemos detenidamente los distintos elementos de formulario que sirven para introducir texto.

El lenguaje HTML nos propone una gran diversidad de alternativas a la hora de crear nuestros formularios, es decir, una gran variedad de elementos para diferentes propósitos. Estas van desde la clásica caja de texto, hasta la lista de opciones en un menú desplegable, pasando por las cajas de validación, etc.

Etiqueta INPUT para texto corto

Las cajas de texto son colocadas por medio de la etiqueta INPUT. Dentro de esta etiqueta hemos de especificar el valor de dos atributos: **type** y **name**.

La etiqueta tendrá la siguiente forma:

```
<input type="text" name="nombre">
```

De este modo expresamos nuestro deseo de crear una caja de texto cuyo contenido será llamado "nombre" (por ejemplo, en el caso de la etiqueta anterior, pero podemos poner distintos nombres a cada uno de los

campos de texto que habrán en los formularios). El aspecto de este tipo de cajas es de sobra conocido, aquí lo podéis ver:

El nombre del elemento del formulario es de gran importancia para poder identificarlo en nuestro programa de procesamiento o en el mail recibido. Por otra parte, es importante indicar el atributo type, ya que, como veremos, existen otras modalidades de elementos de formulario que usan esta misma etiqueta INPUT.

El empleo de estas cajas esta fundamentalmente destinado a la toma de datos breves: palabras o conjuntos de palabras de longitud relativamente corta.

Además de estos dos atributos, esenciales para el correcto funcionamiento de nuestra etiqueta, existen otra serie de atributos que pueden resultarnos de utilidad pero que no son imprescindibles:

size: define el tamaño de la caja de texto, en número de caracteres visibles. Si al escribir el usuario llega al final de la caja, el texto que escriba a continuación también cabrá dentro del campo pero irá desfilando, a medida que se escribe, haciendo desaparecer la parte de texto que queda a la izquierda.

maxlength: indica el tamaño máximo del texto, en número de caracteres, que puede ser escrito en el campo. En caso que el campo de texto tenga definido el atributo maxlength, el navegador no permitirá escribir más caracteres en ese campo que los que hayamos indicado.

value: en algunos casos puede resultarnos interesante asignar un valor definido al campo en cuestión. Esto puede ayudar al usuario a rellenar más rápidamente el formulario o darle alguna idea sobre la naturaleza de datos que se requieren. Este valor inicial del campo puede ser expresado mediante el atributo value. Veamos su efecto con un ejemplo sencillo:

```
<input type="text" name="nombre" value="Perico Palotes">
```

Genera un campo de este estilo:

Veremos posteriormente que este atributo puede resultar bastante relevante en determinadas situaciones.

Hay determinados casos en los que podemos desear esconder el texto escrito en el campo INPUT, por medio asteriscos, de manera que aporte una cierta confidencialidad. Este tipo de campos son análogos a los de

texto, con una sola diferencia: remplazamos el atributo `type="text"` por `type="password"`:

```
<input type="password" name="nombre">
```

En este caso, podéis comprobar que, al escribir dentro del campo, en lugar de texto veréis asteriscos.

Estos campos son ideales para la introducción de datos confidenciales, principalmente códigos de acceso o claves. Se ve en funcionamiento a continuación.



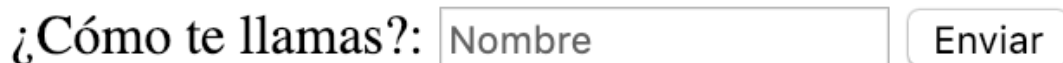
Placeholder

Hasta [HTML5](#) la forma que existía de dar información contextual sobre el contenido que había que insertar en un campo de formulario era mediante el elemento `label`.

En [HTML5](#) aparece el atributo `placeholder`, el cual permite poner información sobre lo que hay que insertar dentro del elemento `input` dentro del mismo elemento.

```
<input type="text" name="nombre" id="nombre" placeholder="Nombre"/>
```

Como podemos ver el texto del `placeholder` queda difuminado dentro del elemento `input`.



Autocomplete

Mediante el atributo `autocomplete` permitiremos al agente de usuario (navegador) que pueda apoyarse en información introducida en otros formularios en el pasado para intentar completar el contenido de un campo ofreciendo alternativas de texto.

Algunos navegadores permiten tener datos preconfigurados para poder autocompletar el contenido del formulario.

```
<input type="text" name="nombre" id="nombre" autocomplete="name">
<input type="text" name="url" id="url" autocomplete="off">
<input type="text" name="direccion" id="direccion" autocomplete="street-address">
```

Los valores que pueden tener pueden ser `on` y `off`. O bien valores que indiquen la semántica del campo para que se pueda autocompletar, estos pueden ser: `name`, `country`, `sex`, `street`, `username`, `organization`,... [puedes consultar más valores para el atributo autocomplete](#)

Autofocus

Hasta [HTML5](#) el control del foco de los elementos `input` de un formulario debía de ser ejecutado mediante código [Javascript](#). Pero ahora ya contamos con un atributo que nos permite indicar cual de los elementos `input` de un formulario empieza a tener el foco del mismo.

Solo uno de los elementos `input` puede tener el atributo `autofocus`. Bastará con añadir el atributo `autofocus` sobre uno de los elementos `input` para que este obtenga el foco cuando se cargue la página.

```
<input type="text" name="url" id="url" autofocus>
```

Nuevos tipos Input

Una de las cosas que nos encontramos en [HTML5](#) son nuevos tipos de elemento de entrada de datos en los formularios. Lo que viene a ser el atributo `type` del elemento `input`.

Hasta [HTML 4.01](#) podíamos definir un elemento `input` como: *text*, *radio*, *checkbox*, *password*, *file*, *hidden* y *submit*.

Recordamos que la estructura de un elemento `input` es:

```
<input type="tipo" id="identificador" size="tamaño" name="nombre" value="texto por defecto"/>
```

Ahora contamos con nuevos tipos de elementos input como son: *search*, *email*, *url*, *tel*, *range*, *number*, *date* y *color*.

Input Search

El tipo `search` de los elementos `input` nos sirven para definir campos de entrada para búsquedas. Es decir, si añadimos un buscador dentro de nuestra web, el campo sobre el cual el usuario podrá introducir el texto de la búsqueda será de tipo `search`.

```
<input type="search" id="busqueda" name="q"/>
```

Vemos que simplemente hemos especificado el valor `search` dentro del atributo `type` del elemento `input`. Al contrario que lo que sucede en otros tipos de elementos `input` en el caso de las búsquedas no hay una visualización específica dentro de los diferentes navegadores.

Input Email

Uno de los datos de contacto más solicitados en Internet es, obviamente, el email. Hasta la versión de [HTML5](#) cuando estamos creando un formulario y queremos que un campo fuese de tipo email lo que hacíamos era declararlo de tipo texto y crearnos unos [códigos javascript de validación de emails](#).

Con la aparición del tipo email, deberemos de marcar el elemento `input` como un elemento del tipo `email`.


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

A partir de este momento **el navegador será el que realice las comprobaciones de que el dato introducido sea un email**. Bien es cierto que, de momento, las validaciones de los navegadores son mínimas y no van más allá del formato del email en cuanto a tener una @ o que después del punto exista un dominio, sea el que fuese.

Visualmente, en *Google Chrome* tenemos el siguiente efecto visual en la validación de un email:

¿Cuál es tu email?



Incluye un signo "@" en la dirección de correo electrónico. La dirección "pepegmail.com" no incluye el signo "@".

De igual manera las opciones que nos ofrece del texto a rellenar serán emails y no cualquier otro tipo de texto. Al menos textos que cuadren con la expresión regular del email.

Input URL

El comportamiento del tipo `url` es parecido al del tipo `email`. La única diferencia es que en este caso **se valida que el contenido insertado coincida con una URL**. Es decir, tenga su protocolo especificado (`http://`), el servidor, el dominio,... con lo cual evitaremos el tener que utilizar, como sucedía anteriormente, validadores de URL.

```
<input type="url" id="website" name="website">
```

Si utilizamos un tipo `url` dentro de un elemento `input` y no insertamos un texto con forma de URL, el navegador nos mostrará lo siguiente:

¿Cuál es la URL de tu web?:



Introduce una URL

Input Tel

El tipo `tel` nos servirá para indicar que el campo es un número de teléfono. Dada la gran cantidad de formatos de números de teléfono que hay en el mundo, no se realizará ninguna validación de formato sobre el campo.

Si bien, la semántica que damos indicando que irá un número de teléfono hace que los campos de tipo `tel` para algunos dispositivos móviles muestre un teclado numérico.

Podremos insertar un campo que gestione un número de teléfono mediante el tipo `tel` de la siguiente forma:

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

Input Number

Los elementos `input` de tipo `number` nos van a valer para poder introducir números. Para crear un elemento `input` de tipo `number` simplemente tendremos que crear la siguiente estructura:

```
<input type="number" id="numero" name="numero">
```

A este tipo de elementos le podemos especificar un par de atributos más. Por un lado podemos fijar cual es el número mínimo y máximo que se puede insertar. Esto lo haremos mediante los atributos `min` y `max`.

```
<input type="number" id="numero" name="numero" min="valor_minimo" max="valor_maximo">
```

Por otro podemos indicar el valor de incremento o decremento de los números. Es decir, si queremos que solo se incremente de dos en dos, o de tres en tres,... Esto lo conseguimos mediante el atributo `step`.

```
<input type="number" id="numero" name="numero" step="incremento">
```

De esta manera podríamos tener el siguiente código fuente:

```
<form action="#" method="get">
  <label for="anio">¿Cuál es tu año par favorito del siglo XXI?</label>
  <input type="number" id="anio" name="anio" min="2000" max="2018" step="2"/>
  <input type="submit" value="Buscar"/>
</form>
```

Vemos que el valor mínimo es *2000* y el máximo *2018* y que solo podemos incrementar o decrementar de 2 en 2, ya que preguntamos por *años pares*.

Los campos `input` de tipo `number` nos forzarán a que el valor introducido sea un número y además que esté dentro del rango marcado por los atributos `min` y `max`.

¿Cuál es tu año par favorito del siglo XXI?



El valor debe ser superior o igual a 2000

Input Range

Otra forma de elegir un valor numérico es mediante un *slider*. Para poder poner estas barras de selección tenemos el tipo `range` dentro de los elementos `input`.

```
<input type="range" id="numero" name="numero">
```

Los elementos `input` de tipo `range` aceptan los mismos parámetros que los elementos `number`. Así que podremos añadir los atributos `min`, `max` y `step`.

```
<input type="range" id="numero" name="numero" min="valor_minimo" max="valor_maximo" step="incremento">
```

De esta manera el mismo ejemplo anterior relativo a los años, pero con un slider quedaría de la siguiente forma:

```
<form action="#" method="get">
  <label for="anio">¿Cuál es tu año par favorito del siglo XXI?</label>
<br/>
  <input type="range" id="anio" name="anio" min="2000" max="2018" step="2"/>
  <input type="submit" value="Enviar"/>
</form>
```

Como se puede ver es idéntico.

Uno de los problemas que nos encontramos con los sliders es que no tienen una representación visual del valor que estamos marcando. Es por ello que deberemos de utilizar algún código en [Javascript](#) para poder mostrarlo.

Así el mismo ejemplo, pero visualizando el año quedaría de la siguiente forma:

```
<form action="#" method="get">
  <label for="anio">¿Cuál es tu año par favorito del siglo XXI?</label>
<br/>
  <input type="range" id="anio" name="anio" min="2000" max="2018" step="2"/>
  <span id="salida"></span><br/>
  <input type="submit" value="Enviar"/>
</form>
```

Y tendríamos que añadir el siguiente código [Javascript](#):

```
var anio = document.getElementById("anio");
var salida = document.getElementById("salida");

salida.textContent = anio.value;
```

```
anio.oninput = function() {  
    salida.textContent = anio.value;  
}
```

Este código [Javascript](#) simplemente vuelca el valor del elemento `input` sobre un elemento `span`.

Input Datetime

Otra de las opciones que han incorporado los elementos de texto `input` en los formularios [HTML5](#) es el manejo de fechas. Para ello contamos con 4 tipos de elementos `input`, que son: `datetime-local`, `month`, `time` y `week`.

Input Datetime-local

Este tipo de elemento nos permite escoger una fecha y hora, sin especificar la zona horaria en la que se encuentra. El valor del tipo `input` será `datetime-local`.

```
<input type="datetime-local" id="fechahora" name="fechahora"/>
```

La representación visual de un tipo `datetime-local` será:

Elige la fecha y la hora:

diciembre de 2018 ▾

lun	mar	mié	jue	vie	sáb	dom
26	27	28	29	30	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	29	30
31	1	2	3	4	5	6

Input Month

En este caso se creará un selector de meses del año. El valor del tipo `input` para los meses en `month`.

```
<input type="month" id="mes" name="mes"/>
```

La representación visual para los campos de texto de tipo mes es:

Elige el mes:

Enviar

diciembre de 2018 ▼

lun	mar	mié	jue	vie	sáb	dom
26	27	28	29	30	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	29	30
31	1	2	3	4	5	6

Input Time

Para poder indicar una hora en formato horas y minutos (*hh:mm*) disponemos del tipo `time` para los elementos de texto de un formulario. La estructura para un elemento `input` de tipo `time` es:

```
<input type="time" id="hora" name="hora"/>
```

Y su representación visual:

Elige la hora:

Podemos comprobar que se le da el formato `*hh:mm*` dentro de la caja de texto.

Input Week

El último elemento para el manejo de fechas es el tipo `week`. En este caso el elemento `week` nos permite seleccionar una semana dentro del año.

```
<input type="week" id="semana" name="semana"/>
```

Vemos que la representación visual cambia y nos muestra las semanas del año dentro del calendario.

Elige la semana:

diciembre de 2018 ▼

Semana	lun	mar	mié	jue	vie	sáb	dom
48	26	27	28	29	30	1	2
49	3	4	5	6	7	8	9
50	10	11	12	13	14	15	16
51	17	18	19	20	21	22	23
52	24	25	26	27	28	29	30
1	31	1	2	3	4	5	6

Input Color

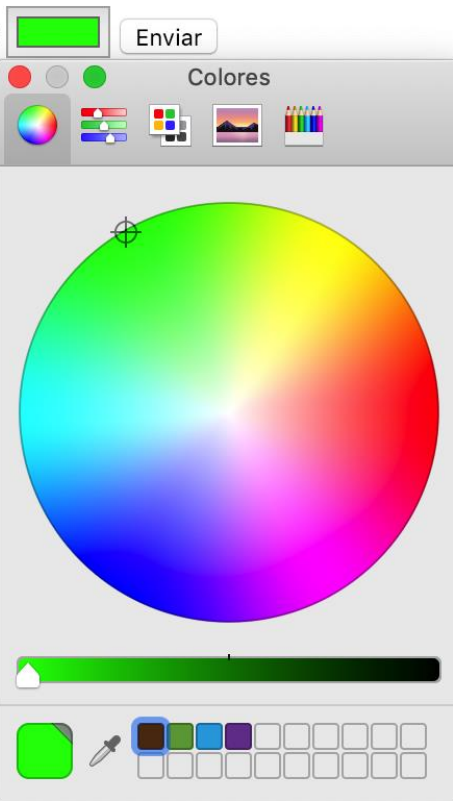
Si lo que queremos es que un usuario nos indique un color de una paleta de colores utilizaremos el elemento `input` con un tipo `color`.

```
<input type="color" name="colorfavorito" value="#ff0000">
```

Mediante el atributo `value` podemos indicar en formato **RGB** un color que será el que muestre la paleta por defecto.

De esta manera, en el navegador, podremos ver algo parecido a lo siguiente:

Elige tu color favorito:



Input File

En [HTML 4.01](#) ya existía un campo de texto `input` en que podíamos subir ficheros. Este era el tipo `file`. En [HTML5](#) sigue existiendo dicho campo, si bien se ha añadido una serie de atributos adicionales para su manejo.

La estructura de un tipo `file` es la siguiente:

```
<input id="ficheros" type="file" name="file" id="file"><br/>
```

Uno de los atributos específicos del tipo `file` que ya existía era el filtrado de tipos de ficheros a aceptar. Esto lo indicamos con el atributo `accept` el cual recibirá una extensión de fichero o un mime-type de los ficheros que aceptamos subir **separados por comas**.

Si queremos hacerlo por extensiones:

```
<input id="ficheros" type="file" name="file" id="file" accept=".jpg,.gif,.png"><br/>
```

O por mimetypes:

```
<input id="ficheros" type="file" name="file" id="file" accept="image/*,audio/*"><br/>
```

En el caso de que queramos subir más de un fichero deberemos de recurrir al atributo `multiple`.

```
<input id="ficheros" type="file" name="file" id="file" accept="image/*" multiple><br/>
```

Por último nos encontramos con el atributo `capture` el cual nos sirve para indicar que el origen del fichero podrá ser la cámara o micrófono del dispositivo. Los valores del atributo `capture` pueden ser **user** para la cámara frontal o **environment** para la cámara trasera.

```
<input id="ficheros" type="file" name="file" id="file" accept="image/*" capture="user"><br/>
```

Etiqueta TEXTAREA para texto largo

Si deseamos poner a la disposición de usuario un campo de texto donde pueda escribir cómodamente sobre un espacio compuesto de varias líneas, hemos de invocar una nueva etiqueta: `TEXTAREA` y su cierre correspondiente.

Este tipo de campos son prácticos cuando el contenido a enviar no es un nombre, teléfono, edad o cualquier otro dato breve, sino más bien, un comentario, opinión, etc. en los que existe la posibilidad que el visitante desee rellenar varias líneas.

Dentro de la etiqueta textarea deberemos indicar, como para el caso visto anteriormente, el atributo name para asociar el contenido a un nombre que será asemejado a una variable en los programas de proceso. Además, podemos definir las dimensiones del campo a partir de los atributos siguientes:

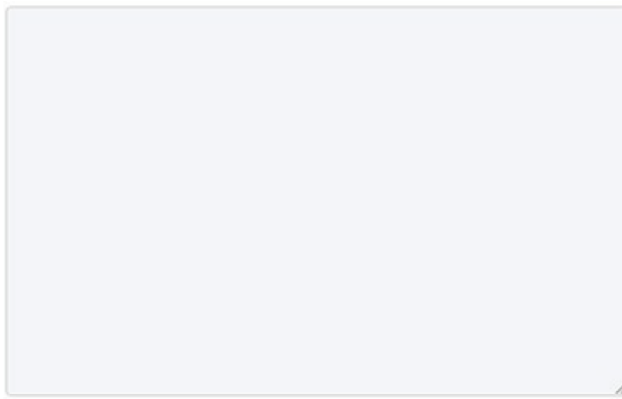
rows: define el número de líneas del campo de texto.

cols: define el número de columnas del campo de texto.

La etiqueta queda por tanto de esta forma:

```
<textarea name="comentario" rows="10"
cols="40"></textarea>
```

El resultado es el siguiente:



Asimismo, es posible predefinir el contenido del campo. Para ello, no usaremos el atributo value, sino que escribiremos dentro de la etiqueta el contenido que deseamos atribuirle.

Veámoslo:

```
<textarea name="comentario" rows="10"
cols="40">Escribe tu comentario....</textarea>
```

Dará como resultado:



Otros elementos de formulario

Explicamos la sintaxis y el funcionamiento de las cajas y listas de selección, casillas de verificación y botones de radio.

Ya hemos explicado sobre la creación de formularios y sobre los [campos de texto](#) en todas sus modalidades. Seguramente hayamos percibido que los textos son una manera muy práctica de hacernos llegar la información del navegante. No obstante, en muchos casos, permitir al usuario que escriba cualquier texto permite demasiada libertad y puede que la información que éste escriba no sea la que nosotros estamos necesitando. Por otra parte, para determinados casos, los textos libres son difícilmente adaptables a programas que puedan procesarlos debidamente. Es por ello que, en determinados casos, puede resultar más efectivo proponer una elección al navegante a partir del planteamiento de una serie de opciones disponibles y definidas por nosotros.

Por ejemplo, pensemos que queremos que el usuario indique su país de residencia. En ese caso podríamos ofrecer una lista de países para que seleccione el que sea. Este mismo caso se puede aplicar a gran variedad de informaciones, como el tipo de tarjeta de crédito para un pago, la puntuación que da a un recurso, si quiere recibir o no un boletín de novedades, etc...

Este tipo de opciones predefinidas por nosotros pueden ser expresadas por medio de diferentes campos de formulario. Veamos a continuación cuales son:

Listas de opciones

Las listas de opciones son ese tipo de menús desplegables que nos permiten elegir una (o varias) de las múltiples opciones que nos proponen. Para construirlas emplearemos una etiqueta `SELECT`, con su respectivo cierre:

Como para los casos ya vistos, dentro de esta etiqueta definiremos su nombre por medio del atributo `name`. Cada opción será incluida en una línea precedida de la etiqueta `OPTION`.

Podemos ver, a partir de estas directivas, la forma más típica y sencilla de esta etiqueta:

```
<select name="estacion">
<option>Primavera</option>
<option>Verano</option>
<option>Otoño</option>
<option>Invierno</option>
```

```
</select>
```

El resultado que obtenemos mediante este código es el que se ilustra en la siguiente imagen:



Esta estructura puede verse modificada principalmente a partir de otros dos atributos:

size: Indica el número de valores mostrados a la vez en la lista. Lo típico es que no se incluya ningún valor en el atributo size, en ese caso tendremos un campo de opciones desplegable, pero si indicamos size aparecerá un campo donde veremos las opciones definidas por size y el resto podrán ser vistos por medio de la barra lateral de desplazamiento.

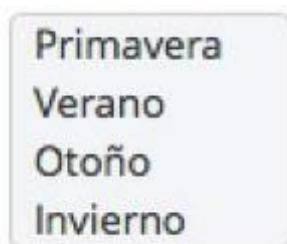
multiple: Permite la selección de más varios elementos de la lista. La elección de más de un elemento se hace como con el explorador de Windows, a partir de las teclas ctrl o mayúsculas (la flecha hacia arriba, también llamada shift). Este atributo se expresa sin valor alguno, es decir, no se utiliza con el igual: simplemente se pone para conseguir el efecto, o no se pone si queremos una lista desplegable común.

```
<select name="estacion">
```

por:

```
<select name="estacion" size="3" multiple>
```

La lista quedara de esta forma:



La etiqueta **OPTION** puede asimismo ser matizada por medio de **otros atributos**

selected: Del mismo modo que multiple, este atributo no toma ningún valor sino que simplemente indica que la opción que lo presenta esta elegida por defecto.

Así, si cambiamos la línea del código anterior:

```
<option>Otoño</option>
```

por:

```
<option selected>Otoño</option>
```

El resultado será:



value: Define el valor de la opción que será enviado al programa o correo electrónico si el usuario elige esa opción. Este atributo puede resultar muy útil si el formulario es enviado a un programa para su procesamiento, puesto que a cada opción se le puede asociar un número o letra, lo cual es más fácilmente manipulable que una palabra o texto. podríamos así escribir líneas del tipo:

```
<option value="1">Primavera</option>
```

De este modo, si el usuario elige primavera, lo que le llegara al programa (o correo) es una variable llamada estacion que tendrá con valor 1. En el correo electrónico recibiríamos:

```
estacion=1
```

Botones de radio

Existe otra alternativa para plantear una elección, en este caso, obligamos al internauta a elegir únicamente una de las opciones que se le proponen.

La etiqueta empleada en este caso es INPUT en la cual tendremos el atributo type ha de tomar el valor radio. Veamos un ejemplo:

```
<input type="radio" name="estacion" value="1">Primavera
```

```
<br>
<input type="radio" name="estacion"
value="2">Verano
<br>
<input type="radio" name="estacion"
value="3">Otoño
<br>
<input type="radio" name="estacion"
value="4">Invierno
```

El resultado es el siguiente:

☐ Primavera ☐ Verano ☐ Otoño ☐ Invierno

Como puede verse, a cada una de las opciones se le atribuye una etiqueta input dentro de la cual asignamos el mismo nombre (name) para todas las opciones y un valor (value) distinto. Si el usuario elige supuestamente Otoño, recibiremos en nuestro correo una línea tal que esta:
estacion=3

Cabe señalar que es posible preseleccionar por defecto una de las opciones. Esto puede ser conseguido por medio del atributo **checked**:

```
<input type="radio" name="estacion" value="2"
checked>Verano
```

Veamos el efecto:

☐ Primavera ☒ Verano ☐ Otoño ☐ Invierno

Cajas de validación

Este tipo de elementos pueden ser activados o desactivados por el visitante por un simple clic sobre la caja en cuestión. La sintaxis utilizada es muy similar a las vistas anteriormente:

```
<input type="checkbox" name="paella">Me gusta la
paella
```

El efecto:

☐ Me gusta la paella

La única diferencia fundamental es el valor adoptado por el atributo type. Del mismo modo que para los botones de radio, podemos activar la caja por medio del atributo **checked**.

El tipo de información que llegara a nuestro correo (o al programa) será del tipo: paella=on (u off dependiendo si ha sido activada o no).

Envío, borrado y demás en formularios HTML

Enseñamos la manera de colocar botones de envío y borrado en formularios HTML. También conocemos los campos invisibles y los botones normales. Además, hacemos un ejemplo práctico.

Siguiendo con la explicación de todo lo relativo a formularios, ha llegado el momento de explicar cómo podemos hacer un botón para provocar el envío del formulario, entre otras cosas.

Como podremos imaginarnos, en formularios no solamente habrá elementos o campos donde solicitar información del usuario, sino también habrá que implementar otra serie de funciones.

Concretamente, han de permitirnos su envío mediante un botón. También puede resultar práctico poder proponer un botón de borrado o bien acompañar el formulario de datos ocultos que puedan ayudarnos en su procesamiento.

Botón de envío de formulario (botón de submit)

Para dar por finalizado el proceso de relleno del formulario y hacerlo llegar a su gestor, el navegante ha de enviarlo por medio de un botón previsto a tal efecto. La construcción de dicho botón no reviste ninguna dificultad una vez familiarizados con las etiquetas INPUT ya vistas:

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

Con este código generamos un botón como este:



Como puede verse, tan solo hemos de especificar que se trata de un botón de envío (type="submit") y hemos de definir el mensaje que queremos que aparezca escrito en el botón por medio del atributo value. Este tipo de

campos INPUT, para envío de formularios, a menudo se conocen simplemente como "botones de submit".

Botón de borrado (botón de reset)

Este botón nos permitirá borrar el formulario por completo, en el caso de que el usuario desee rehacerlo desde el principio. Su estructura sintáctica es análoga a la anterior:

```
<input type="reset" value="Borrar">
```

A diferencia del botón de envío, indispensable en cualquier formulario, el botón de borrado resulta meramente optativo y no es utilizado frecuentemente. Hay que tener cuidado de no ponerlo muy cerca del botón de envío y de distinguir claramente el uno del otro, para que ningún usuario borre el contenido del formulario que acaba de rellenar por error.

Datos ocultos (campos hidden)

En algunos casos, aparte de los propios datos rellenos por el usuario, puede resultar práctico enviar datos definidos por nosotros mismos que ayuden al programa en su procesamiento del formulario. Este tipo de datos, que no se muestran en la página pero sí pueden ser detectados solicitando el código fuente, no son frecuentemente utilizados por páginas construidas en HTML, son más bien usados por páginas que emplean tecnologías de servidor. No os asustéis, veremos más adelante qué quiere decir esto. Tan solo queremos dar constancia de su existencia y de su modo creación. He aquí un ejemplo:

```
<input type=hidden name="sitio"  
value="www.desarrolloweb.com">
```

Esta etiqueta, incluida dentro de nuestro formulario, enviara un dato adicional al correo o programa encargado de la gestión del formulario. Podríamos, a partir de este dato, dar a conocer al programa el origen del formulario o algún tipo de acción a llevar a cabo (una redirección por ejemplo).

Botones normales

Dentro de los formularios también podemos colocar botones normales, pulsables como cualquier otro botón. Igual que ocurre con los campos hidden, estos botones por sí solos no tienen mucha utilidad pero podremos necesitarlos para realizar acciones en el futuro. Su sintaxis es la siguiente.

```
<input type=button value="Texto escrito en el  
botón">
```

Quedaría de esta manera:

Texto escrito en el botón

El uso más frecuente de un botón es en la programación en el cliente. Utilizando lenguajes como **Javascript** podemos definir acciones a tomar cuando un visitante pulse el botón de una página web.

Ejemplo completo de formulario

Con este capítulo finalizamos el tema de formularios. Pasemos ahora a ejemplificar todo lo aprendido a partir de la creación de un formulario que consulta el grado de satisfacción de los usuarios de una línea de autobuses ficticia. El formulario está construido para que envíe los datos por correo electrónico a un buzón determinado.

Vemos el formulario en esta página. Vosotros tratar de construirlo para ver si habéis entendido bien los temas sobre formularios.

The image shows a web form with the following elements:

- Nombre**: A text input field.
- Email**: A text input field with an '@' symbol as a placeholder.
- Población**: A text input field.
- Sexo**: Radio buttons for 'Hombre' (selected) and 'Mujer'.
- Frecuencia de los viajes**: A dropdown menu showing 'Varias veces al dia'.
- Comentarios**: A label followed by a large text area for comments.
- Footer**: A checked checkbox for 'Deseo recibir notificación de las novedades en las líneas de autobuses.', and two buttons: 'Enviar formulario' and 'Borrar todo'.

El código del formulario se puede ver a continuación. Pero antes de analizarlo te recomendamos construir el formulario por tu propia cuenta para practicar.

```
<form action="mailto:colabora@desarrolloweb.com"
method="post" enctype="text/plain">
```

```

Nombre <input type="text" name="nombre" size="30"
maxlength="100">
<br>
Email <input type="text" name="email" size="25"
maxlength="100" value="@">
<br>
Población <input type="text" name="poblacion" size="20"
maxlength="60">
<br>
Sexo
<br>
<input type="radio" name="sexo" value="Varon" checked>
Hombre
<br>
<input type="radio" name="sexo" value="Hembra"> Mujer
<br>
<br>
Frecuencia de los viajes
<br>
<select name="utilizacion">
<option value="1">Varias veces al dia
<option value="2">Una vez al dia
<option value="3">Varias veces a la semana
<option value="4">varias veces al mes
</select>
<br>
<br>
Comentarios sobre su satisfacción personal
<br>
<textarea cols="30" rows="7" name="comentarios"></textarea>
<br>
<br>
<input type="checkbox" name="recibir_info" checked> Deseo
recibir notificación de las novedades en las líneas de
autobuses.
<br>
<br>
<input type="submit" value="Enviar formulario">
<br>
<br>
<input type="Reset" value="Borrar todo">
</form>

```

Etiquetas FIELDSET y LEGEND de formularios

Las etiquetas de HTML FIELDSET y LEGEND sirven para crear bloques de elementos dentro de formularios.

Los atributos FIELDSET y LEGEND se utilizan en conjunto y sirven respectivamente para definir y etiquetar grupos lógicos de elementos de formularios. Realmente no afectan a la operativa del formulario, pero

sirven para agrupar elementos en diferentes áreas, de modo que se clarifique la entrada de datos del usuario. Al formar varios grupos de elementos se puede crear una estructura mucho más fácil de asimilar por el usuario, sobre todo si se trata de formularios que tengan muchos elementos.

Etiqueta FIELDSET

La etiqueta FIELDSET sirve para agrupar los elementos. Se utiliza con su respectiva etiqueta de cierre y lo que hace es crear un recuadro que rodea a los elementos de formulario colocados dentro de ella.

Por ejemplo, se podría usar de esta manera:

```
<fieldset>
Elemento de formulario: <input type="text"
name="elemento1">
<br>
Otro elemento: <input type="text" name="otro">
</fieldset>
```

Simplemente creará un cuadrado que agrupará los dos elementos del formulario incluidos dentro del FIELDSET.

Etiqueta LEGEND

LEGEND sirve para nombrar o etiquetar un grupo creado con FIELDSET. Añade simplemente una nota aclaratoria sobre qué tipo de información se está agrupando en el recuadro. Tampoco sirve para nada en especial, de no ser porque queda bonita y porque puede servir para ayudar al usuario y mejorar la interfaz y la claridad de los formularios.

La etiqueta LEGEND se coloca después de la etiqueta FIELDSET. Tiene su propia etiqueta de cierre. Entre LEGEND y su cierre colocamos el texto con el que queremos marcar el recuadro definido con FIELDSET.

A la etiqueta LEGEND se le puede poner el atributo align para indicar el lugar donde debe aparecer la leyenda. Por ejemplo podríamos indicar align="right" para que apareciera en la parte de la derecha, en lugar de la izquierda, que es donde aparece por defecto.

Veamos ahora un ejemplo sencillo de utilización de las etiquetas FIELDSET y LEGEND en conjunto.

```
<form>
<fieldset>
<legend align="right">Datos personales</legend>
Nombre: <input type="text" name="nombre">
<br>
Edad: <input type="text" name="edad" size="2">
<br>
```

```

Dirección: <input type="text" name="direccion">
</fieldset>
<br>
<fieldset>
<legend align="right">Datos de tu
ordenador</legend>
Modelo de ordenador: <input type="text"
name="modelo">
<br>
Sistema que te da el problema:
<select>
<option value=cpu>CPU
<option value=impresora>Impresora
</select>
</fieldset>
<br>
<fieldset>
<legend align="right">Descripción del
problema</legend>
<textarea cols="55" rows="8"
name="descripcion"></textarea>
</fieldset>
</form>

```

Para ver el resultado, recomendamos pegar el código al editor de código. Podremos comprobar como aparecen tres bloques en el formulario, producidos por tres etiquetas FIELDSET, con varios campos de formulario incluidos en cada una. Además, cada uno de los FIELDSET tienen dentro un LEGEND que sirve para nombrar con una leyenda cada uno de los tres bloques.

Etiqueta LABEL

Aunque no forma parte del objetivo de este artículo, queremos nombrar también otra etiqueta llamada LABEL que sí tiene una utilidad especial en la creación de formularios, además de la estética. Sirve para poner texto al lado de los elementos de formulario y que tal texto esté asociado al propio elemento. Ese texto, que pondremos con el tag LABEL, se asocia a un elemento concreto con el atributo FOR, colocando como valor del atributo el identificador del campo que se está asociando.

```

<label for="edad">Edad</label> <input type="text"
name="edad" id="edad">

```

Como vemos, hemos creado un LABEL y hemos colocado en el atributo FOR el nombre del campo de formulario que estamos asociando a ese

texto. El resultado es que el texto colocado dentro de LABEL es un elemento interactivo, al que podemos hacer clic y sería como si hiciésemos clic en el propio campo asociado al LABEL.

LABEL

Hasta no hace mucho los textos que ponemos al lado de los campos de formulario no estaban asociados a dichos campos. Es decir, el texto que colocamos al lado de un elemento de formulario, para especificar qué debe escribir el usuario en el campo, no tiene ninguna relación real con el propio elemento de formulario.

Por ejemplo, si tenemos un código como este:

```
Dirección: <input type="text" name="direccion">
```

El texto "Dirección" no está asociado para nada con el campo INPUT. Por ello, al pulsar sobre el texto "Dirección" no ocurre nada. Esto es así también con otros campos de formulario, como las cajas de checkbox o botones de radio.

```
<input type="checkbox" name="interesado"> Estoy  
interesado
```

Si pulsamos sobre el texto que hay al lado del campo de confirmación "Estoy interesado", no sucede nada! Pero ahora, con la utilización de la etiqueta LABEL podemos conseguir que, haciendo clic en el texto "Estoy interesado", el control checkbox cambie de estado.

Ejemplo:

```
<form action="#" method="post"  
enctype="text/plain" name="un ejemplo más">  
<label>  
<input type="checkbox" name="email">  
Recibir email  
</label>  
</form>
```

Ese ejemplo de LABEL es perfectamente válido y asocia el texto "Recibir email" al campo checkbox de formulario, de manera que si pulsamos sobre "Recibir email" cambiará el estado del campo checkbox asociado. Sin embargo, en la etiqueta LABEL podemos utilizar un atributo llamado FOR, que sirve para indicar explícitamente a qué campo de formulario se está asociando ese texto. Para ello colocamos como valor del atributo FOR el identificador del campo que estamos asociando a ese LABEL. Esto nos permite una mayor versatilidad a la hora de componer el HTML de nuestra página. Veamos el siguiente ejemplo:

```
<form>
```

```

<label for="hombre">Hombre</label>
<input type="radio" name="sexo" id="hombre"
value="hombre">
<br>
<label for="mujer">Mujer</label>
<input type="radio" name="sexo" id="mujer"
value="mujer">
</form>

```

Si ponemos este ejemplo en marcha, veremos que pulsando en el texto "Hombre" se activa el botón de radio "hombre". Del mismo modo, si pulsamos sobre el texto "Mujer" se activará la opción del radio button "mujer". Podemos ver cómo quedaría ese código en marcha a continuación, aunque es una imagen, no podrás hacerlo funcionar:

Hombre



Mujer



BUTTON

Esta etiqueta proporciona un método único para la implementación de cualquier tipo de botón de formulario. Sus principales atributos son: type="tipo", que puede tomar los ya conocidos valores submit (por defecto), reset y button.

name="nombre", que asigna un nombre identificador único al botón.

value="texto", que define el texto que va a aparecer en el botón.

La principal ventaja que aporta estas etiquetas es que ahora vamos a poder introducir dentro de ellas cualquier elemento de HTML, como imágenes y tablas.

Podemos ver un ejemplo a continuación.

```

<form action="#" method="post"
enctype="text/plain" name="miform">
<button name="boton_1" type="button">
<table width="10" cellpadding="2"
border="1">
<tr>
<td>uno</td>
<td>dos</td>
</tr>
<tr>

```

```
<td>tres</td>
<td>cuatro</td>
</tr>
</table>
</button>
</form>
```