

## **UF1304: Elaboración de plantillas y formularios**

# Presentación

## Identificación del Módulo o unidad formativa

Bienvenido a la Unidad Formativa **UF1304: Elaboración de plantillas y formularios**. Esta unidad formativa pertenece al Módulo Formativo **MF0950\_2: Construcción de páginas web** del Certificado de Profesionalidad **IFCD0110 Confección y publicación de páginas web** que pertenece a la familia profesional de Informática y comunicaciones.

## Presentación de los contenidos

La finalidad de esta unidad formativa es adquirir los conocimientos necesarios para poder elaborar y administrar un formulario en un sitio web, conociendo en profundidad su utilidad, sus distintos elementos y métodos de control de la información. Además en esta unidad formativa el alumno verá lo que es una plantilla web, sus diversos elementos y usos, finalmente aprendiendo a crearlas y administrarlas.

## Objetivos

Al finalizar esta unidad formativa aprenderás a:

- Confeccionar plantillas para las páginas web atendiendo a las especificaciones de diseño recibidas.
- Crear formularios e integrarlos en páginas web para incluir interactividad en las mismas, siguiendo unas especificaciones funcionales recibidas.

# Índice

UD1. Formularios en la construcción de páginas web.....	7
1.1. Características.....	9
1.1.1. La interactividad de las páginas web.....	12
1.1.2. La variabilidad de los datos de la página web.....	15
1.1.3. El envío de información a servidores .....	19
1.2. Elementos y atributos de formulario.....	23
1.2.1. Descripción y definición de los elementos de un formulario.....	91
1.2.2. Utilización de campos y textos.....	94
1.2.3. Etiquetas de los formularios .....	100
1.2.4. Tamaños, columnas y filas de los formularios.....	116
1.3. Controles de formulario .....	128
1.3.1. Descripción de los controles de los formularios .....	132
1.3.2. Utilización de botones de acción .....	137
1.3.3. Utilización de lista desplegables.....	166
1.3.4. Utilización de casillas de verificación .....	184
1.3.5. Utilización de campos de texto .....	207
1.4. Formularios y eventos. Criterios de accesibilidad y usabilidad en el diseño de formularios.....	242

## UF1304: Elaboración de plantillas y formularios

1.4.1. Agrupación de datos .....	246
1.4.2. Adecuación del tamaño del formulario (división en ..... distintas páginas) .....	257
1.4.3. Identificación de los campos obligatorios.....	261
1.4.4. Ordenación lógica de la petición de datos.....	265
1.4.5. Información correcta al usuario. ....	270
1.4.6. Utilización de páginas de error y de confirmación.....	274
<b>UD2. Plantillas en la construcción de páginas web .....</b>	<b>285</b>
2.1. Funciones y características .....	287
2.1.1. Descripción de una plantilla web.....	290
2.1.2. Elementos de una plantilla web.....	294
2.1.3. Estructura y organización de los elementos de las ..... plantillas.....	296
2.1.4. Especificar las zonas modificables de una plantilla y las partes fijas .....	297
2.1.5. Utilización de plantillas .....	301
2.2. Campos editables y no editables .....	318
2.2.1. Definir y crear los campos susceptibles de cambios en una plantilla .....	321
2.2.2. Definir y crear los campos no modificables en una .... plantilla .....	322
2.3. Aplicar plantillas a páginas web .....	324
2.3.1. Las plantillas en la web.....	338
2.3.2. Búsqueda de plantillas en la red .....	341
2.3.3. Adaptación de plantillas a páginas web.....	345
<b>Glosario .....</b>	<b>363</b>
<b>Soluciones .....</b>	<b>367</b>

# UD1

Formularios en la  
construcción de  
páginas web

## UF1304: Elaboración de plantillas y formularios

## 1.1. Características

- 1.1.1. La interactividad de las páginas web
- 1.1.2. La variabilidad de los datos de la página web
- 1.1.3. El envío de información a servidores

## 1.2. Elementos y atributos de formulario

- 1.2.1. Descripción y definición de los elementos de un formulario
- 1.2.2. Utilización de campos y textos
- 1.2.3. Etiquetas de los formularios
- 1.2.4. Tamaños, columnas y filas de los formularios

## 1.3. Controles de formulario

- 1.3.1. Descripción de los controles de los formularios
- 1.3.2. Utilización de botones de acción
- 1.3.3. Utilización de lista desplegables
- 1.3.4. Utilización de casillas de verificación
- 1.3.5. Utilización de campos de texto

## 1.4. Formularios y eventos. Criterios de accesibilidad y usabilidad en el diseño de formularios

- 1.4.1. Agrupación de datos
- 1.4.2. Adecuación del tamaño del formulario (división en distintas páginas).
- 1.4.3. Identificación de los campos obligatorios
- 1.4.4. Ordenación lógica de la petición de datos
- 1.4.5. Información correcta al usuario
- 1.4.6. Utilización de páginas de error y de confirmación

## 1.1. Características

Cómo ya sabéis una página web es, simplemente, un documento adecuado para los navegadores web World Wide Web, para que dicho navegador muestre dicha página web en un monitor o dispositivo que deseemos visualizarla.

Sabías que el antiguo navegador web Netscape, anterior incluso a Internet Explorer, fue el origen, al liberar su código, de uno de los navegadores webs más conocidos y utilizados en la actualidad, el Mozilla Firefox..

La página web es lo que se muestra, pero no hay que olvidar que todo ello está alojado y programado en un archivo en un ordenador, generalmente escrito en HTML o en los múltiples lenguajes de programación que tenemos hoy día a nuestra disposición, cuya principal función es brindar unos hipertextos que además permiten navegar a otras páginas web a través de enlaces.

- En el primer nivel tenemos los ordenadores locales, es aquí donde se conectan los usuarios, que pueden estar en un hogar, una oficina, un ciber café, etc. Hoy día incluso los smart phones entrarían en esta categoría.
- A continuación tenemos el resto de recursos de la red local. Si tenemos una impresora conectada en red, un switch para distribuir la señal entre los ordenadores que están a más bajo nivel, etc.
- A continuación tenemos el router, al que bien se puede acceder de forma física, o a través de una red inalámbrica. Este router es el que gestiona todos los paquetes que entran y salen desde y hacia internet.
- Tras el router, y con el paso intermedio del ISP (Proveedor de Servicios de Internet) se llega a la gran red de redes, internet. En ella, gracias a una serie de protocolos de enrutamiento, los paquetes llegan a su destino pudiendo recorrer en décimas de segundo varias líneas de todo el mundo.
- En el destino espera el servidor, que es el que recibe la señal, pasando por todos los pasos similares de los ordenadores locales con los usu-

rios. Tras recibir el conjunto de paquetes completo enviado, lo juntará y tras procesar la información, lo volverá a enviar a su destino, pasando por el mismo camino de vuelta.

Además, los navegadores web coordinan los recursos web que utilizamos en la página tal y como esté programada, tales como hojas de estilo, scripts e imágenes, para presentar la página web.

Sabías que aunque los orígenes de internet se remontan a 1969, en realidad el protocolo World Wide Web, que es lo que la mayoría de personas reconocen hoy en día como internet, empezó a crearse en 1989, y no se empezó a utilizar hasta 1991.

Inicialmente las páginas webs eran estáticas, lo que quería decir que eran mostradas exactamente como se almacenaban en el sistema de archivos del servidor web.

Poco a poco se acabó derivando hacia las páginas web dinámicas, generadas por una aplicación web que es controlada por el software del servidor o de scripting en el lado del mismo usuario.

En esta evolución del diseño web, y en la necesidad de recoger los datos introducidos por los usuarios para crear bases de datos o una mayor interactividad, se hizo evidente la necesidad de los formularios web en los lenguajes básicos de hipertexto.

Un formulario (formulario web o formulario HTML) en una página web que permite que un usuario introduzca unos datos con un formato preestablecido que se envían a un servidor para su procesamiento.



Los datos que el usuario introduce deben ser enviados a un servidor para su procesamiento, o como mínimo, enviarse a algún lugar lógico donde puedan ser procesados adecuadamente.

---

Los formularios originalmente se llamaron así por su similitud a los formularios comunes de papel, del tipo de llenar los datos personales.

## UD1

Finalmente se han adaptado a las bases de datos para que los usuarios de internet rellenen los formularios con cajas de texto, botones de opciones o casillas de selección.

Por ejemplo, los formularios se pueden utilizar para introducir datos personales para suscribirse a una revista online, o se pueden utilizar para contactar con el servicio técnico de una web de venta de productos que ni siquiera los oferta por internet pero sí tienen página web, correo electrónico u otros medios donde poder recoger la información.



Los primeros formularios web, por supuesto, estaban programados en HTML (lenguaje de marcas de hipertexto). Que cómo ya sabéis es un lenguaje estándar para la elaboración de páginas web, que se basa en poner el código entre etiquetas escritas entre paréntesis angulares (como <html>), siendo la primera etiqueta en una pareja la de inicio, y la segunda la de cierre.

Hoy en día ya pueden encontrarse en muchos otros lenguajes, como son Perl, Java o .NET. Incluso se pueden programar en lenguajes de script como JavaScript, o incluso en arquitecturas cerradas con lenguajes de múltiples usos como SAP.

Como veremos en esta unidad didáctica, todos estos formularios tienen elementos y características comunes, que serán los que tengamos que programar y administrar al crearlos.

Los formularios permiten que el usuario introduzca datos en el servidor para que sean procesados, lo que permite crear páginas web mucho más interactivas y variables, y sobre todo, recoger información que el usuario de la web tendrá que introducir por sí mismo. Estas son las características más importantes de un formulario y deben tenerse siempre presentes.

### 1.1.1. La interactividad de las páginas web

El uso de formularios en las páginas webs introdujo un concepto, que seguramente le resulte conocido al alumno, pero que tuvo gran importancia en el desarrollo inicial de la world wide web, como es el de la interactividad.

Sencillamente, se puede decir que la interactividad es la capacidad paulatina y cambiante que tiene un medio (ya sea de información, de comunicación o de relación interpersonal) para darle a los usuarios un mayor manejo tanto en la selección e interacción con los contenidos, así como en las posibilidades de comunicación y expresión.

Se podría resumir en que, la interactividad es la capacitación del receptor de un mensaje, que está directamente relacionado con él, hasta el grado que establezca el emisor, dentro de los límites que tenga dicha comunicación.

**Importante:** La interactividad al principio del desarrollo informático y de comunicaciones supuso un esfuerzo y un trabajo muy importante para poder planificar una correcta navegación entre los distintos textos, imágenes y vínculos, para que el usuario realmente sintiese que controlaba y estaba al mando de las aplicaciones que utilizaba. Por eso, lo que hoy día puede suponer algo trivial, fue en realidad todo un reto de ingenio y de diseño en los albores de la comunicación informática.

Se puede decir que en realidad hay varios niveles en lo que a interactividad se refiere, pero básicamente podemos decir que en informática, la interactividad es simplemente la capacidad de interacción entre la máquina o servidor y el propio usuario que está accediendo a ese servicio.

Niveles de interactividad.	
No interactivo	Cuando un elemento no se relaciona con elementos anteriores.
Reactivo	Cuando un elemento se relaciona únicamente con un elemento inmediatamente anterior.
Interactivo	Cuando un elemento está relacionado con un número de elementos anteriores y existe una relación entre ellos

Las páginas webs siempre fueron, en su inicio y por definición, estáticas.

Se podía afirmar que no eran completamente no interactivas, porque tenían mensajes o textos que se relacionaban con otros previos. Pero en realidad tampoco llegaban a ser completamente interactivas, ya que se relacionaban únicamente con un enlace previo inmediato.

## UD1

Dicho de otro modo, una página web simplemente era un conjunto de información a la que accedía el usuario para poder leerla, pero no podía modificarla o aportar datos propios.

Esto se solucionó en parte gracias a la creación de formularios web, como veremos en esta unidad didáctica, pero la evolución en el sistema no se quedó en eso.



Hoy día el uso de páginas webs dinámicas, cada vez más interactivas con el usuario, están a la orden del día.

Se puede preguntar a la mayoría de la población, y muchos afirmarían que la característica que ellos destacarían de internet, por encima del resto, es la interactividad.

Es cierto, que ningún otro medio conocido hasta ahora ha ofrecido a los usuarios un nivel de interactividad tan alto. Los usuarios que leían un periódico, escuchaban la radio o veían la tele siempre tenían la opción de cambiar de canal, pero esto en realidad es un nivel de interactividad tan baja, que se puede considerar prácticamente ridículo comparado con lo que ofrece internet.

De hecho hoy en día, muchas televisiones, viendo el auge que tiene internet y su interactividad, están estudiando introducir, o lo han hecho ya en algunos casos, servicios interactivos como guías de programación, teletextos mejorados, o incluso (mediante el uso de discos duros internos y routers para poder comunicarse con el exterior) poder votar en reality shows como Gran Hermano o participar en chats con otros telespectadores.



Esta interactividad no está marcada, como se piensa muchas veces erróneamente, con el desarrollo de las aplicaciones multimedia en la web, sino más bien con el desarrollo de herramientas sociales y a la evolución de la transmisión de datos e información entre las propias páginas web y los usuarios.

Los formularios pasaron de ser una herramienta para enviar datos al servidor para su procesamiento, y hoy en día han evolucionado junto con las bases de datos a la creación de encuestas, comentarios en blogs, compras online de productos, etc.



**Una página web dinámica** es aquella que genera su contenido en función de los datos que introduce un usuario a través de una web o un formulario.

---

Al punto ha llegado la web hoy día, que con las redes sociales y los smart phones, las posibilidades de interacción, no solo entre el usuario y la web, si no entre los mismos usuarios, tiene más importancia que nunca.

De hecho, se ha desarrollado una interactividad con internet tan intensa entre los propios internautas y las herramientas que se ofrecen hoy día, que se puede decir que el usuario cada día es más experto, buscando la información que necesita más rápida y eficazmente, y teniendo la capacidad de discriminar la información que no necesita.

## UD1



Hasta el nacimiento de internet la publicidad era siempre unidireccional. Los carteles publicitarios, los anuncios en revistas o periódicos, las campañas publicitarias en radio o televisión, etc. Sin embargo, hoy en día la comunicación tiene claramente dos direcciones, del emisor al receptor y viceversa. De hecho, también se puede afirmar que hay una dirección del emisor a otro emisor, lo cual hace que incluso los propios intermediarios se pierdan.

### 1.1.2. La variabilidad de los datos de la página web

Llegados a este punto, el alumno debería comprender la importancia de una web dinámica ante una web estática. Además, el alumno debería entender las bases de la función de un formulario en una página web. En todo caso, debemos entrar en uno de los puntos más importantes asociados a los formularios web, que es la propia variabilidad dinámica de los datos en las páginas webs.

Para ello, vamos a ver el formulario quizás más conocido, y a la vez más desapercibido de internet en la actualidad. Un formulario que se usa cientos de millones de veces al día, que no es otro que el buscador de Google.



Esa pequeña caja de texto junto con dos botones asociadas en su parte inferior no es, ni más ni menos, que un formulario web, pero que sirve para, mediante herramientas de programación propias, desarrollar un algoritmo de búsqueda de información relacionada con lo que el usuario haya introducido interactivamente en la caja de texto.



Google tiene registrados varios dominios que son muy parecidos al verdadero cambiando alguna letra, ya que muchos usuarios escriben mal la palabra "google" en sus navegadores, y que todos estos dominios te llevan al sitio web correcto. Algunos ejemplos son google.com, google.com, goolge.com, etc.

Con este formulario se ve un ejemplo sencillo de los datos que puede mostrar una página web, y de su variabilidad, en función de los datos introducidos por un usuario en un momento dado, pero se usan miles de ejemplos cada día.

Por ejemplo, al registrarse en una página web normalmente tenemos que introducir nuestros datos a través de un formulario, y una vez registrados, para iniciar sesión también usaremos un formulario.

Por supuesto, los datos mostrados en una página web normalmente varían si el usuario ha iniciado sesión y está registrado, que si no lo está.

Otro ejemplo lo tenemos en las webs de compras online, según se vayan usando los distintos buscadores, que son pequeños formularios, se van mostrando productos distintos. Pero es que además, al comprar una lista de productos es necesario introducir los datos personales como método de pago, dirección donde se realiza el envío, etc.

## UD1

La relevancia de los formularios en la elaboración de las páginas webs es evidente, ya que gracias a ellos la interactividad, y sobre todo la variabilidad de las mismas según la información introducida por el usuario, es una herramienta vital en el desarrollo web.



Una página web, gracias a los formularios, se convierte en un entorno abierto y cambiante, mostrando unos datos que varían en función de lo que introduzca el usuario, que establece de esta forma una comunicación con el servidor de forma totalmente opaca hacia él, pero administrada de forma totalmente transparente de cara al administrador del sitio web.

Para terminar de hacernos una idea de la importancia de los formularios web y de la variabilidad de internet según estos, vamos a ver dos ejemplos:

- El formulario de alta en una red social como facebook. Sin la existencia de los formularios, seguramente no podrían existir esta, ni ninguna otra red social. Pero lo que es más importante, imagina lo que sería si no hubiese que llenar el formulario para ingresar en el facebook y, cualquier que pudiese acceder a tu ordenador, o incluso a tu red local a través de tu router, pudiese acceder a tus fotos personales o hacerse pasar por ti.

La imagen muestra una captura de pantalla de la página web de Facebook en el navegador Mozilla Firefox. Se pueden ver claramente dos formularios. El de arriba a la derecha es el formulario para ingresar en la página web, el de abajo a la derecha es el formulario de registro, el que, además de tener más campos y ser más completo, tiene una función y una llamada a un algoritmo totalmente diferente en el servidor de destino.



Prácticamente uno de cada trece habitantes del planeta tiene una cuenta de facebook, por lo que es seguro que cada uno de ellos tuvo que llenar, como mínimo, el formulario de registro en su web. Además, que existe un formulario en facebook para denunciar la existencia de una cuenta de una persona fallecida. Así que, tanto para entrar en facebook, como para salir, se debe llenar un formulario web.

- El formulario de búsqueda de un producto en una página web de compras o de subastas. Sin estos formularios habría que recorrer páginas y páginas para intentar encontrar el objeto que quisiésemos comprar. Por muy bien organizadas que estuviesen las páginas web, en sitios como ebay, sería prácticamente imposible encontrar lo que se busca ya que hay millones de objetos que se subastan cada día.

## UD1

La imagen muestra una captura de pantalla de la página web de Ebay en el navegador Mozilla Firefox. Se pueden ver el formulario de búsqueda de objetos en la parte superior de la página en la mitad. Sin este buscador sería prácticamente imposible encontrar un objeto concreto que se quisiese comprar mediante una subasta en esta página.



Ebay se vende de todo, incluido patatas, papel higiénico, sábanas, piedras, etc. Incluso se puede llegar a ver objetos tan extravagantes como un calcetín sin su pareja, una uña de un pie o un clip doblado. Si a pesar de todo, quisieras comprar uno de esos objetos tan singulares, si no existiese el formulario de búsqueda en la página las posibilidades de encontrar el objeto serían muy próximas a cero.

---

### 1.1.3. El envío de información a servidores

Como ya sabemos, cuando un usuario rellena y envía un formulario, realmente lo que está haciendo es enviar unos datos o información desde su máquina hasta el servidor web.

Esto lo hace usando una serie de elementos de entrada que veremos en profundidad más adelante en esta unidad didáctica, pero se puede decir que el flujo de esta información sigue siempre unos pasos específicos:

- El usuario introduce los datos usando los distintos elementos del formulario, y a continuación pulsa el botón designado para enviarlos.
- El navegador que esté usando el usuario manda esos datos al servidor que se haya indicado en el formulario.
- El servidor recibe los datos enviados, procesándolos, y enviando el resultado obtenido de vuelta al navegador del usuario.
- El navegador del usuario le muestra a este el resultado.

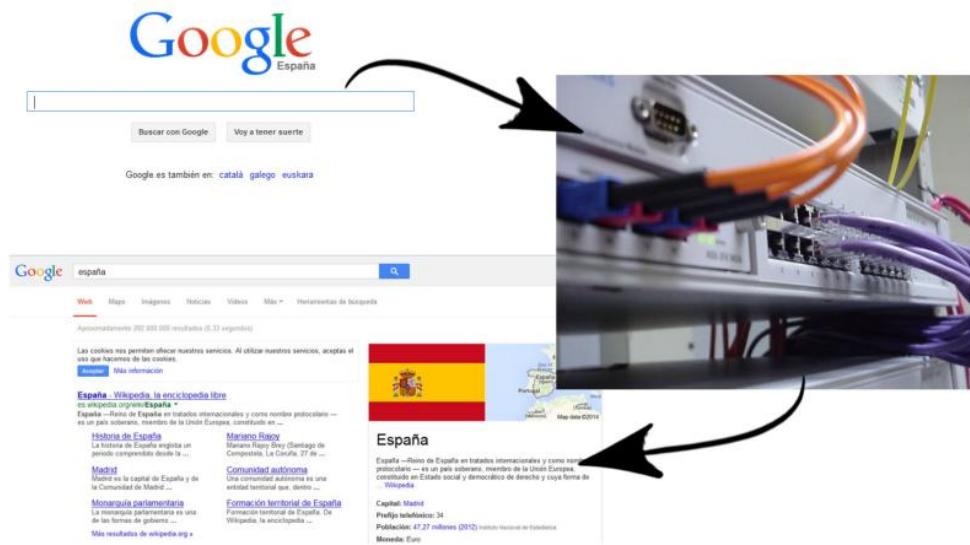


Imagen del flujo que sigue la información al llenar y enviar un formulario. Primero el usuario introduce los datos, estos cuando se envían van a un servidor, que cuando los procesa devuelve una página web asociada a los datos introducidos por el usuario, que será mostrada en el navegador de este.



El primer servidor, que se creó en 1981, fue el IMB VM Machine, que permitió la colaboración por correo electrónico para grupos de correo, ya tuvo que soportar spam (correo basura no solicitado, generalmente con publicidad) y los primeros trols de internet (persona que manda mensajes provocadores o totalmente fuera de lugar con la intención de molestar o sacar de quicio al resto de usuarios de un servicio).

---

En realidad lo que manda normalmente el navegador al servidor, al introducir los datos el usuario y darle al botón de enviar, son el nombre de cada elemento del formulario con el valor asociado que le ha introducido el propio usuario.

Pero el servidor puede devolver elementos más complejos, ya que el lenguaje de programación con el que trabaja es totalmente opaco al usuario, e incluso puede ser independiente del lenguaje utilizado para enviar los datos del propio formulario.

## UD1



Aunque los datos enviados por un formulario suelen ser procesados por lenguajes en el propio servidor, existen scripts para la modificación dinámica de una web en el lado del cliente, como JavaScript.

A la hora de programar un formulario, el programador debe ser totalmente consciente de cuál es su función, y sobre todo de dónde se van a enviar los datos para su procesamiento.

Además, como veremos en esta unidad didáctica, se pueden elegir varios métodos para enviar esta información, dependiendo de diversos factores:

- El tamaño del formulario.
- Si se va a trabajar con una base de datos o no
- Si el formulario tiene información privada o relevante, como passwords, datos personales, etc.

Por ejemplo, no es lo mismo encontrarse con un formulario como el siguiente:

The screenshot shows a registration form for 'Cinelania'. The fields include:

- Nickname:** Input field with placeholder 'Ejige tu nick de usuario: caracteres alfanuméricos y [ - \_ ]' and a 'Comprobar Nick' button.
- Email:** Input field for email address.
- Confirm Email:** Input field for confirming the email address.
- Password:** Input field for password.
- Confirm Password:** Input field for confirming the password.
- Country:** Select dropdown menu set to 'España'.
- Community / Region:** Select dropdown menu set to '(Selecciona tu región)'.
- Preferred Language:** Radio buttons for 'Español' and 'Inglés'.
- Sex:** Select dropdown menu set to 'Hombre'.
- Date of Birth:** Input fields for day, month, and year.
- Checkboxes:**
  - Checkboxes for receiving newsletters from 'miarroba.es' and 'Recibir promociones y consejos por email'.
  - Checkboxes for accepting terms and conditions and agreeing to receive automatic emails.
- Captcha:** CAPTCHA image showing 'B T K Z'.
- Buttons:** 'Continuar ...' (Continue) button and a link 'Contactar con el administrador de esta comunidad'.

## UF1304: Elaboración de plantillas y formularios

La captura de pantalla muestra un formulario de acceso a un foro de miarroba, una comunidad de foros que gozó de gran popularidad hace una década, pero que tienen un sistema de programación algo obsoleto en la actualidad. Sin embargo su sistema de formularios sigue siendo correcto y totalmente seguros a la hora de enviar datos relevantes del usuario, como por ejemplo en este caso la contraseña o sus datos personales.

Que con un formulario como el siguiente:



Captura de pantalla de la web de envío de noticias en español más popular en internet, menéame.net, en el navegador Mozilla Firefox. A la derecha en su parte superior se puede ver un formulario de búsqueda, destinado a buscar noticias específicas en la web según la palabra que se introduzca en la caja de texto.

En el primer formulario, tanto el volumen del formulario, como el almacenamiento de la información (que se va a hacer en una base de datos donde se guardará encriptada la contraseña), como la relevancia de los datos enviados, hacen que su programación deba ser totalmente distinta.

Además que, es presumible que el acceso al formulario de búsqueda a la web de noticias deba ser más rápido, no solo porque pida información de forma inmediata, sino también porque será usado cientos de veces al minuto y tenga que soportar un volumen de tráfico mucho mayor que cualquier formulario de alta en la web.

En todo caso, el método elegido debe ser el más adecuado para la seguridad y los lenguajes de programación que vayan a ser utilizados para utilizar estos datos, y sobre todo debe tenerse en cuenta la cantidad de información que va a ser enviada mediante el formulario.

Algunos formularios permiten, incluso, el envío de archivos al servidor.

## UD1

En estos formularios deberá especificarse el tipo de codificación empleada al enviarlo al servidor, y además a la hora de programarlo, se tendrá que tener muy en cuenta el ancho de banda y la cantidad de tráfico que puede soportar el servidor sobre el que se está programando dicha aplicación, ya que una planificación ineficiente de estos conceptos puede lastrar el rendimiento de toda la página web en la que está el formulario.

### 1.2. Elementos y atributos de formulario

Los formularios se pueden crear en código HTML mediante el uso de una sencilla etiqueta: <form>.

Esta etiqueta debe tener, a su vez, unos distintos atributos para definir los distintos valores del formulario en sí, y unos elementos dentro del formulario para representar los datos que el usuario puede introducir y, más tarde, enviar al servidor.

Para definir los elementos dentro de un formulario se utiliza la etiqueta <input>, que dependiendo de los atributos que se le asignen permitirá definir varios tipos de elementos, como son botones, cuadros de texto, casillas de verificación, etc.



Importante

Las dos etiquetas más importantes para la elaboración de formularios en código HTML son la etiqueta <form> que inicia y define los atributos del formulario, y la etiqueta <input> que inicia y define los elementos del formulario.

---

Ejemplo del código de un formulario básico. En él se pueden ver claramente las etiquetas <form> para definir los atributos del formulario, y dos etiquetas <input>, para definir dos elementos del formulario, un cuadro de texto y un botón de enviar:

```
<html>
<head>
<title>Ejemplo de formulario simple.</title>
```

```

</head>

<body>

    <form action="http://www.paginadeejemplo.com/formulario.php" method="post">

        Nombre: <input type="text" name="nombre" value="" />
        <br/>

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

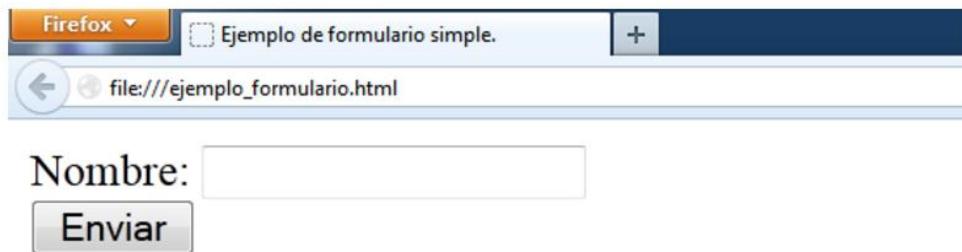
    </form>

</body>

</html>

```

Y el resultado, una vez se ejecute este código debe ser el siguiente:

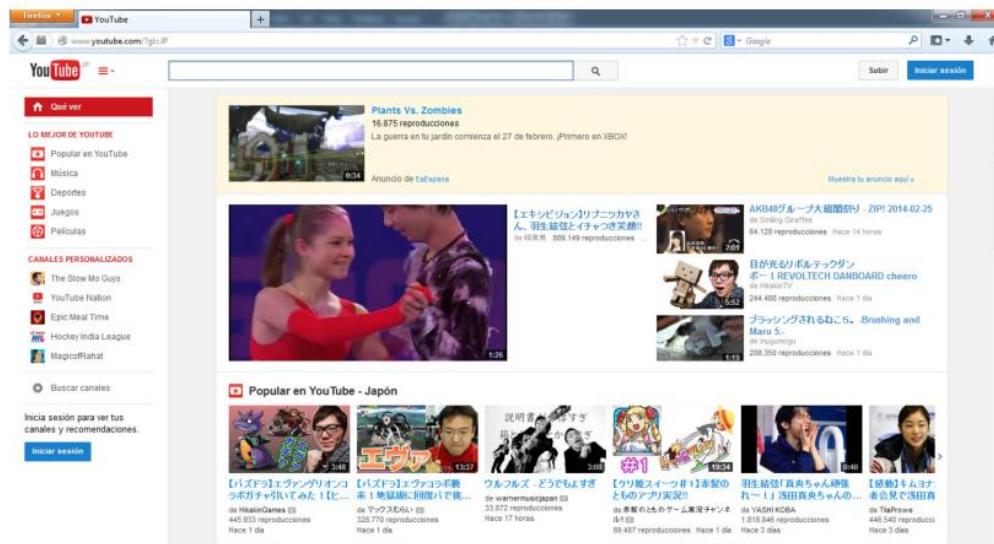


A partir de este momento van a sucederse en la Unidad Formativa gran cantidad de líneas de código con la intención de que el alumno se familiarice, y además aprenda de forma escalonada, el código que necesita para aprender a realizar formularios. Pero, además, el alumno debe realizar estos programas él mismo, programándolos en su propio ordenador, y nombrándolos como la unidad en curso. Por ejemplo, este código de arriba debería llamarse el archivo "formulario.1.2.html".

## UD1

En la etiqueta <form> se pueden usar varios atributos. En concreto, como en la mayoría de etiquetas HTML se pueden usar los atributos comunes como son:

- Los atributos básicos para definir identificadores a los elementos del código o asociar las clases CSS al elemento.
- Los atributos internacionales para indicar el idioma del elemento.
- O incluso los atributos de eventos si la página está usando JavaScript para realizar acciones dinámicas en los elementos de la página.
- Y los atributos de foco, relacionados en casi todos los casos con la accesibilidad de los sitios web.



Captura de pantalla de la web de videos más utilizada en internet, youtube, en el navegador Mozilla Firefox. Esta web tan internacional está programada para que soporte prácticamente todos los idiomas más utilizados en el mundo. En este ejemplo vemos una captura de pantalla de su versión japonesa.

Por supuesto, todos estos atributos son muy importantes en algunos casos en los formularios, y no deben perderse de vista en ningún momento.

Sin embargo en esta unidad didáctica vamos a fijarnos en los atributos únicos y más importantes de la etiqueta <form>, que son los atributos:

Atributo	Función
action	Indica la URL donde se envían los datos del formulario.
method	Especifica el método HTTP que se utiliza al enviar el formulario.
enctype	Tipo de codificación que se utiliza para enviar los datos al servidor.
accept	Indica los archivos que el servidor acepta, cuando se le envían archivos.
name	Especifica el nombre del formulario.
target	Señala donde mostrar la respuesta que se reciba tras enviar el formulario.
accept-charset	Especifica la codificación de caracteres que se van a usar para el envío del formulario.
autocomplete	Activa o desactiva la función de autocompletar del formulario.
novalidate	Atributo que si está presente indica que los datos del formulario no deben ser validados cuando se envían.

### Atributo action

Todos los formularios requieren, obligatoriamente, que el atributo "action" esté presente y sea correcto, ya que si no es así, los datos del formulario no llegarían a su destino, y el resto de la programación del mismo no serviría para nada. La URL especificada en este atributo puede ser de dos tipos: absoluta o relativa.

- Si la URL es absoluta, tiene la ventaja de que no importa el nivel en el que se encuentre el formulario, o que incluso el destino donde se envíen los datos sea una página web distinta. Simplemente hay que introducir la dirección completa y cuando el formulario se envíe ese será su destino.
- Si la URL es relativa, esto quiere decir que el destino del envío del formulario se encuentra en la misma página web, y que sólo tendremos que poner la dirección del archivo en una carpeta dentro de esa misma página, ahorrándonos el paso de tener que buscar la dirección absoluta en internet.

## UD1

En cualquier caso si la URL especificada no es válida, el formulario dará un error al ser enviado.

Veamos algunos ejemplos.

A continuación vemos el código HTML de un formulario con una URL absoluta

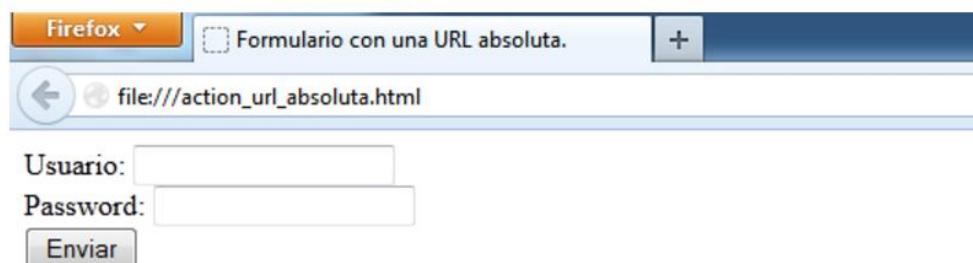
```

<html>
  <head>
    <title>Formulario con una URL absoluta.</title>
  </head>
  <body>
    <form action="http://www.páginadeejemplo.com/formulario.php" method="post">
      Usuario: <input type="text" name="usuario" value="" />
      <br/>

      Password: <input type="password" name="contrasena" value="" /> <br/>

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

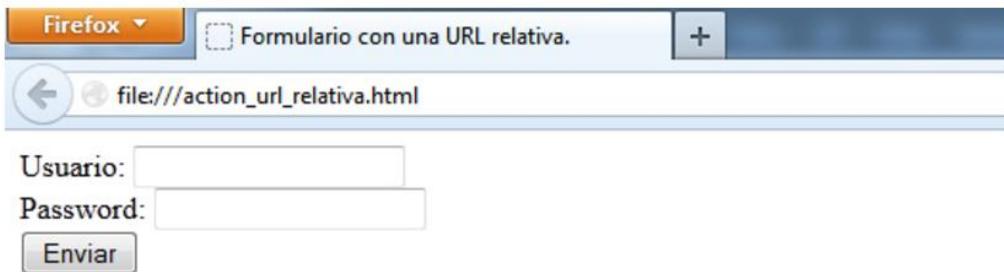
Ahora vemos una captura de pantalla del resultado que debe dar el código al ejecutarlo en un navegador web.



A continuación veremos el código HTML de un formulario con una URL relativa.

```
<html>
<head>
    <title>Formulario con una URL relativa.</title>
</head>
<body>
    <form action="formulario.php" method="post">
        Usuario: <input type="text" name="usuario" value="" />
        <br/>
        Password: <input type="password" name="contrasena" value="" />
        <br/>
        <input type="submit" value="Enviar" />
    </form>
</body>
</html>
```

Veamos una captura de pantalla del resultado que debe dar el código al ejecutarlo en un navegador web.



Véase que aunque el código no difiere prácticamente en nada de un ejemplo a otro, en realidad el resultado es bastante distinto a la hora de ejecutarse en una web.

## UD1

En cualquier caso, la programación del atributo action, ya sea expresando la URL de forma absoluta o relativa, responde más a una función meramente de diseño funcional más que de diseño visual, ya que es prácticamente opaco de cara al usuario.

Por ejemplo, en el caso de una web donde prima la velocidad y los usuarios no disponen, de media, de una conexión muy avanzada, lo más recomendable suele ser optar por URLs relativas.

En el caso de que los usuarios dispongan de una buena conexión, o en webs con mucho tráfico y con grandes caudales para absorberlo, se puede optar por una URL absoluta.

En la actualidad esta última vía suele ser la más utilizada, aunque en formularios más modestos nunca se debe descartar la anterior.



Importante

Aunque el atributo "action" es obligatorio en la etiqueta <form> en HTML para saber donde se van a enviar los datos del formulario, en HTML5 se puede programar esta misma funcionalidad en el botón de enviar el formulario, por lo que este atributo no sería necesario en este caso específico.

---

### Atributo method

El atributo "method" se utiliza para indicar el método HTTP utilizado para enviar el formulario, y puede tomar dos valores: "get" o "post".

- El valor "get" envía los datos como variables al final de la URL especificada en el atributo action, al final del mismo.

Este método es ideal para formularios pequeños, como formularios de búsqueda en una web. También es el más adecuado si se quieren enviar datos con un formulario que no deben ser añadidos o borrados de una base de datos.

Aquí vemos el código de un ejemplo del atributo method con el valor "get".

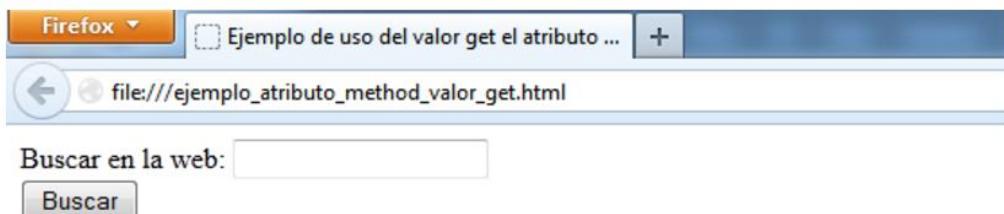
```
<html>
```

```

<head>
    <title> Ejemplo de uso del valor get el atributo method en un formulario.</title>
</head>
<body>
    <form action="busqueda.php" method="get">
        Buscar en la web:
        <input type="text" name="buscar" value="" />
        <br/>
        <input type="submit" value="Buscar" />
    </form>
</body>
</html>

```

Captura de pantalla del resultado que debe dar el código al ejecutarlo en un navegador web.



Si se ejecuta el código de este formulario, se puede ver en la URL cómo intenta enviar la información al lugar indicado en el atributo "action", con el valor al final de la URL introducido a través del atributo "method".

- El valor "post" envía los datos como cabeceras HTTP, por lo que los datos enviados en el formulario no aparecen en la URL.

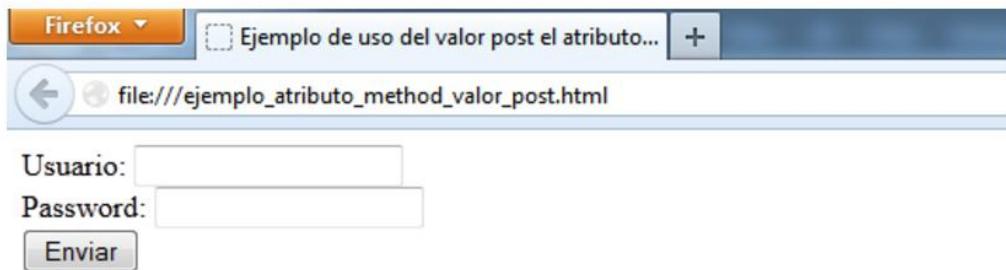
## UD1

Este método es ideal para formularios grandes, si se quiere enviar un archivo junto con el formulario, si este contiene datos privados o confidenciales (como contraseñas, por ejemplo) y si se quiere añadir o eliminar datos de una base de datos.

Aquí vemos el código de un ejemplo del atributo method con el valor "post".

```
<html>
<head>
    <title>Ejemplo de uso del valor post el atributo method en un formulario.</title>
</head>
<body>
    <form action="ingreso.php" method="post">
        Usuario:
        <input type="text" name="usuario" value="" />
        <br/>
        Password:
        <input type="password" name="contrasena" value="" />
        <br/>
        <input type="submit" value="Enviar" />
    </form>
</body>
</html>
```

Captura de pantalla del resultado que debe dar el código al ejecutarlo en un navegador web.



Si se ejecuta el código de este formulario, se puede ver en la URL cómo intenta enviar la información al lugar indicado en el atributo "action", pero no se verán en la URL los valores introducidos a través del atributo "method".

Las especificaciones del lenguaje HTML técnicamente define la diferencia entre "get" y "post", específicamente como el sistema que se usa para codificar (por un navegador) los datos en una URL, que como ya sabemos la diferencia es que aparezcan o no los datos del formulario en el cuerpo del mensaje.

Sin embargo, las especificaciones también dan la recomendación de usar el método "get" cuando el procesamiento de formularios es "idempotente" (la propiedad para realizar una acción determinada varias veces y aún así conseguir el mismo resultado que se obtendría si se realizase una sola vez), y en esos casos solamente.

Como simplificación, podríamos decir que "get" se utiliza simplemente para recuperar los datos mientras que "post" puede implicar cualquier cosa, como el almacenamiento o la actualización de los datos, comprar un producto, el envío de un correo electrónico, etc. Si el servicio asociado con el procesamiento de un formulario tiene efectos secundarios, como por ejemplo, la modificación de una base de datos o la suscripción a un servicio, el método debe ser siempre "post".

De todas maneras, se pueden decir que las especificaciones del protocolo HTTP no son muy claras al respecto. Y se puede decir que los usuarios no pueden ser considerados responsables de los efectos secundarios de la implementación de un método u otro, lo que presumiblemente significa que el programador debe elegir el método que más se ajuste al proceso actual.

En concreto, se ha establecido que los métodos "get" y "head" no deben usarse, o al menos no ser relevantes, para realizar una acción que no sea la recuperación. Estos métodos no deben ser considerados seguros.

## UD1

A pesar de todas las precauciones que tome el programador, se debe pensar que cualquier método que se utilice para enviar datos es susceptible a ser crackeado usando software malicioso. Por lo que el envío de información importante debe ser reducido hasta un mínimo lo más razonable posible.



Sabías qué

Ni la NASA o el ejército de los Estados Unidos, a pesar de toda la seguridad de sus sistemas, se han librado de la vulneración de sus sistemas. En 2001, un hacker llamado Gary McKinnon entró a sus sistemas realizando el hackeo más grande de la historia militar de todos los tiempos.

Uso y características de los dos valores del atributo method	
get	Añade los datos enviados del formulario al final de la URL.
	La longitud de la URL es limitada (3.000 caracteres)
	No se debe usar para enviar datos importantes o privados, como contraseñas.
	Es también útil para formularios donde un usuario quiere visualizar los resultados tras enviarlos.
	Es más rápido y efectivo para envío de formularios con datos no importantes.
post	Los datos enviados no van añadidos a la URL, si no en el código que se envía.
	No tiene limitación de tamaño.
	Los usuarios no pueden visualizar los resultados después de enviarlos.



Importante

Si no se especifica, el valor por defecto del atributo "method" es "get".

### Atributo enctype

Este atributo es utilizado para indicar el tipo de codificación que se quiere usar al enviar el formulario al servidor. Este método se debe utilizar en los formularios que permiten adjuntar archivos.



Importante

Este atributo puede ser utilizado solamente si se usa el valor "post" en el atributo "method" para enviar el formulario.

Valores que puede tomar el atributo enctype	
application/x-www-form-urlencoded	Todos los caracteres son codificados antes de ser enviados (los espacios son enviados como "+" y los caracteres especiales son convertidos a ASCII hexadecimal).
multipart/form-data	Los caracteres no son codificados.
text/plain	Los espacios son enviados como "+", pero el resto de caracteres no son codificados.



Importante

Si no se especifica, el valor por defecto del atributo "enctype" es "application/x-www-form-urlencoded".

Aquí vemos el código de un ejemplo del atributo enctype.

```
<html>
<head>
    <title>Ejemplo de uso del atributo enctype en un formulario.</title>
</head>
```

## UD1

```

<body>

    <form action="envio_de_archivo.php" method="post"
enctype="multipart/form-data">

        Manda el archivo:

        <input type="file" name="archivo" />
        <br/>
        <input type="submit" value="Mandar Archivo"/>
    </form>

</body>

</html>

```

Captura de pantalla del resultado que debe dar el código al ejecutarlo en un navegador web.



Pero también se puede mandar el archivo con otra codificación:

```

<html>

    <head>

        <title>Ejemplo de uso del atributo enctype en un for-
mulario.</title>

    </head>

    <body>

        <form action="envio_de_archivo.php" method="post"
enctype="application/x-www-form-urlencoded">

```

**Usuario:**

```
<input type="text" size="30" maxlength="30"
name="usuario" value="Usuario" />
```

```
<br/>
```

```
<br/>
```

**Password:**

```
<input type="password" size="10" maxlength="10"
name="password" />
```

```
<br/>
```

**Confirmar password:**

```
<input type="password" size="10" maxlength="10"
name="c_password" />
```

```
<br/>
```

```
<br/>
```

**Manda el archivo:**

```
<input type="file" name="archivo" />
```

```
<br/>
```

```
<br/>
```

```
<input type="submit" value="Mandar Archivo"/>
```

```
</form>
```

```
</body>
```

```
</html>
```

O se puede mandar con esta otra codificación:

```
<html>
```

```
<head>
```

```
  <title>Ejemplo de uso del atributo enctype en un formulario.</title>
```

# UD1

```

</head>

<body>

    <form    action="envio_de_archivo.php"    method="post"
enctype="text/plain">

        Usuario:

        <input    type="text"    size="30"    maxlength="30"
name="usuario" value="Usuario" />

        <br/>
        <br/>

        Password:

        <input    type="password"    size="10"    maxlength="10"
name="password" />

        <br/>

        Confirmar password:

        <input    type="password"    size="10"    maxlength="10"
name="c_password" />

        <br/>
        <br/>

        Manda el archivo:

        <input type="file" name="archivo" />

        <br/>
        <br/>

        <input type="submit" value="Mandar Archivo"/>

    </form>

</body>

</html>

```

## Atributo accept

Este atributo es utilizado cuando los formularios permiten adjuntar archivos, para indicar todos los tipos de archivos aceptados por el servidor, separados por comas.

Los tipos especificados utilizados deben estar dentro de unos estándares que se pueden encontrar bajo el nombre de MYME TYPES. Por ejemplo, si queremos subir tan sólo imágenes gif y jpeg, podría usarse la siguiente sintaxis:

```
<form action="envio_imagenes.php" accept="image/gif,image/jpeg">
```



No se debe utilizar este atributo como una herramienta de validación de archivos, esa función debe ser realizada en el servidor.

---

Aquí vemos el código de un ejemplo del atributo accept.

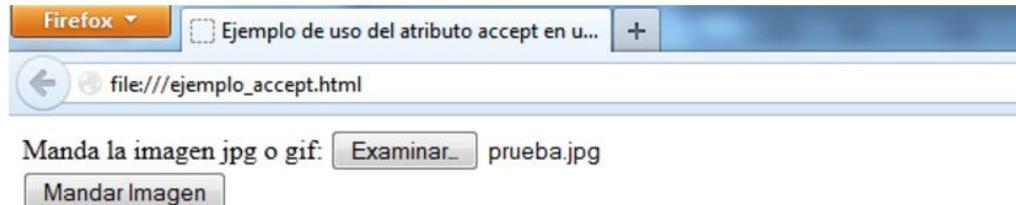
```
<html>
<head>
    <title>Ejemplo de uso del atributo accept en un formulario.</title>
</head>
<body>
    <form      action="envio_imagenes.php"      method="post"
accept="image/gif,image/jpeg">
        Manda la imagen jpg o gif:
        <input type="file" name="imagen" />
        <br/>
        <input type="submit" value="Mandar Imagen"/>
    </form>

```

## UD1

```
</body>
</html>
```

Captura de pantalla del resultado que debe dar el código al ejecutarlo en un navegador web.



### Atributo name

Este atributo se utiliza para especificar el nombre de un formulario.

Este nombre se puede utilizar, por ejemplo, para ser usado como referencia para elementos en JavaScript, o simplemente para utilizarlo como referencia del formulario una vez se han enviado los datos.

Se puede usar el atributo común "id" en lugar del atributo "name", ya que su función es similar y este atributo también sirve para ser utilizado como referencia en el caso de que se utilice como referencia para scripts.

Sin embargo, el atributo "name" es propio de la etiqueta "form".

Los valores que se pueden introducir a este atributo es cualquier texto plano que el programador quiera asociar con el formulario.



En XHTML se desaconseja el uso de este atributo, por lo que es recomendable utilizar el atributo común "id".

Aquí vemos el código de un ejemplo del atributo name.

```
<html>

<head>

    <title>Ejemplo de uso del atributo name en un formulario.</title>

</head>

<body>

    <form action="formulario.php" method="post" name="form_envio_nombre">

        Nombre:

        <input type="text" name="nombre" value="" />

        <br/>

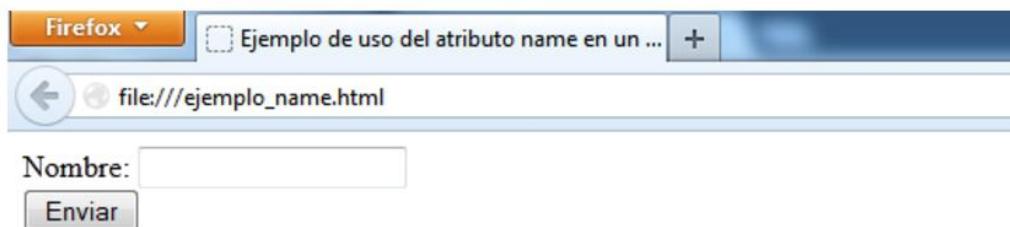
        <input type="submit" value="Enviar" />

    </form>

</body>

</html>
```

Captura de pantalla del resultado que debe dar el código al ejecutarlo en un navegador web.



## UD1

### Atributo target

Este atributo es utilizado cuando el programador quiere que la web resultante que devuelva el servidor tras haber enviado los datos del formulario el usuario se muestre en una nueva ventana o marco de la ventana del navegador.

Valores que puede tomar el atributo target	
_blank	El resultado se muestra en una nueva ventana o pestaña.
_self	El resultado se muestra en la misma ventana que el formulario enviado.
_parent	El resultado se muestra en el marco "padre" del actual.
_top	El resultado se muestra a pantalla completa en la ventana.
nombre del marco	El resultado se muestra en el marco del nombre que se haya introducido (que se hará con texto plano).



Si no se especifica, el valor por defecto del atributo "target" es "\_self".

Además, los valores "\_parent", "\_top" y "nombre del marco" sólo pueden ser utilizados en lenguajes y entornos donde se usen los marcos.

Aquí vemos el código de un ejemplo del atributo target.

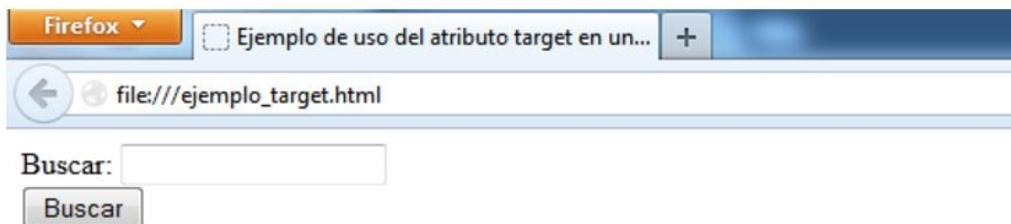
```
<html>
  <head>
    <title>Ejemplo de uso del atributo target en un formulario.</title>
  </head>
  <body>
    <form action="resultado_busqueda.php" method="post"
      target="_blank">
      Buscar:
    </form>
  </body>
</html>
```

```

<input type="text" name="buscar" value="" />
<br/>
<input type="submit" value="Buscar" />
</form>
</body>
</html>

```

Captura de pantalla del resultado que debe dar el código al ejecutarlo en un navegador web.



## Atributo accept-charset

Este atributo es utilizado para especificar que codificación van a llevar los caracteres del formulario cuando se envíe al servidor, y al igual que ocurre en el atributo "accept", pueden utilizarse varios valores separados por comas, o en este caso también se pueden separar por espacios.

El valor por defecto de este atributo es la cadena "UNKNOWN", que indica que la codificación con la que se envía el documento es la misma que la que contiene el formulario.

Las codificaciones más utilizadas son:

- UTF-8: Desarrollado por el Consorcio Unicode, es su estándar de codificación de caracteres que contiene todos los símbolos de puntuación, caracteres y símbolos escritos del mundo.
- ISO-8859-1: Que es una extensión del código ASCII con caracteres internacionales añadidos, siendo más completo que ANSI.

## UD1



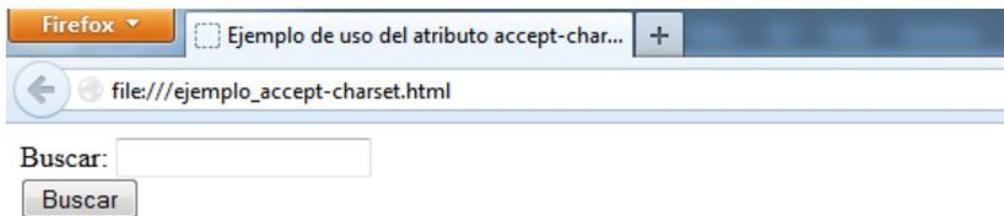
El término Unicode, que se ha utilizado para crear el estándar de caracteres internacional, proviene de los tres objetivos perseguidos cuando se creó en octubre de 1991: universalidad, uniformidad y unicidad.

---

Aquí vemos el código de un ejemplo del atributo accept-charset.

```
<html>
  <head>
    <title>Ejemplo de uso del atributo accept-charset en
    un formulario.</title>
  </head>
  <body>
    <form action="resultado_busqueda.php" method="post"
      accept-charset="ISO-8859-1">
      Buscar:
      <input type="text" name="buscar" value="" />
      <br/>
      <input type="submit" value="Buscar" />
    </form>
  </body>
</html>
```

Captura de pantalla del resultado que debe dar el código al ejecutarlo en un navegador web.



## Atributo autocomplete

Esta reciente implementación a los formularios permite que, si el navegador lo permite y está configurado para ello, los campos del formulario se autocompletan basándose en valores que el usuario ha introducido anteriormente.

Esta función hace que, por ejemplo, si dentro una misma página web (o incluso en sitios web distintos) hay varios formularios con campos similares, el usuario si tiene activada la función autocompletar puede elegir datos rellenos con anterioridad.

Los posibles valores de este atributo son:

- On: Que permite que se active la función de autocompletar en el formulario.
- Off: Que desactiva la función de autocompletar del usuario.



Si no se especifica, el valor por defecto del atributo "autocomplete" es "on".

Además, es posible que el formulario tenga la función de autocompletar activada, y algunos campos específicos la tengan desactivada, y viceversa.

---

Aquí vemos el código de un ejemplo del atributo autocomplete.

```
<html>
<head>
```

## UD1

```

<title>Ejemplo de uso del atributo autocomplete en un formulario.</title>

</head>

<body>

    <form action="resultado_busqueda.php" method="post"
autocomplete="off">

        Buscar:

        <input type="text" name="buscar" value="" />

        <br/>

        <input type="submit" value="Buscar" />

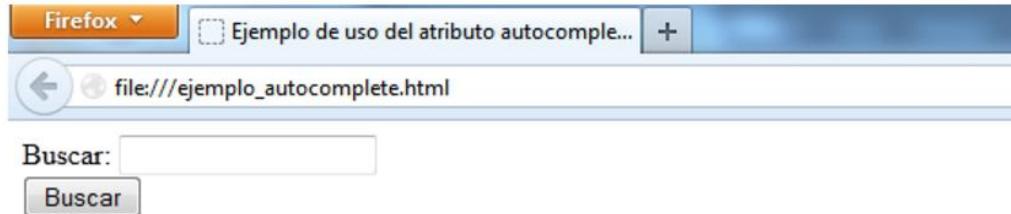
    </form>

</body>

</html>

```

Captura de pantalla del resultado que debe dar el código al ejecutarlo en un navegador web.



### Atributo novalidate

Este atributo es en realidad una variable booleana.

Su función específica es que, si está declarado en el formulario, se considera que es true y por lo tanto los datos del formulario no se deben validar durante la presentación del formulario.



El atributo novalidate es un atributo vacío. Esto significa que se puede escribir simplemente la palabra "novalidate" en el código y ese formulario no será validado por el navegador. Aunque en XHTML se puede escribir novalidate = "novalidate".

---

Aquí vemos el código de un ejemplo del atributo autocomplete.

```
<html>
<head>
    <title>Ejemplo de uso del atributo novalidate en un formulario.</title>
</head>
<body>
    <form action="envio_archivo.php" method="post" novalidate>
        Manda el archivo:
        <input type="file" name="archivo" />
        <br/>
        <input type="submit" value="Mandar Archivo"/>
    </form>
</body>
</html>
```

Captura de pantalla del resultado que debe dar el código al ejecutarlo en un navegador web.

## UD1



Hay que saber diferenciar entre los atributos “novalidate” y “formnovalidate”. El primero se aplica al formulario, evitando que sea validado. El “formnovalidate” se aplica a un botón de envío, lo que significa que el formulario no se valida si se usa ese botón.

Por ejemplo, cuando un usuario está guardando datos de un formulario en lugar de publicarlos estos pueden ser incompletos y no válidos, pero como no se requiere validación para ser salvados se podría usar para ese caso.

Y con estos atributos básicos, podemos definir casi todos los formularios escritos en lenguaje HTML, pero también utilizados en otros lenguajes de cada al servidor como PHP.

Aquí vemos el código de un ejemplo de todos los atributos explicados en un formulario de alta en una página web.

```

<html>
  <head>
    <title>Ejemplo de uso de los atributos de method en un formulario.</title>
  </head>
  <body>
    <form action=>>formulario_de_alta.php<> method=>>post<>
      enctype=>>application/x-www-form-urlencoded<>
      accept=>>image/jpeg<> name=>>formulario_de_prueba<> target=>>_blank<> accept-charset=>>UTF-8<> autocomplete=>>off<>
      novalidate>
        Usuario:
        <input type=>>text<> name=>>usuario<> value=>>> />
    </form>
  </body>
</html>

```

```

<br/>

    Password: <input type=>password</input> name=>contrasena</input> va-
    lue=>>> />

<br/>

    Repetir Password: <input type=>password</input>
    name=>confirmacion_contrasena</input> value=>>> />

<br/>

    Subir avatar de usuario:

<input type=>file</input> name=>avatar</input> />

<br/>

<input type=>submit</input> value=>Enviar</input> />

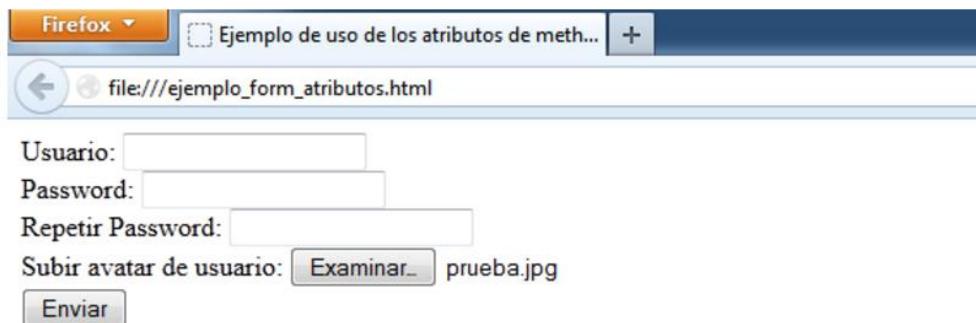
</form>

</body>

</html>

```

Captura de pantalla del resultado que debe dar el código al ejecutarlo en un navegador web.



### Atributos Comunes Aplicados a los Formularios

Además de los atributos generales explicados anteriormente, el alumno debe tener capacidad para usar los atributos comunes aplicados en los formularios que diseñe.

## UD1

Muchos de ellos son de uso habitual en el código HTML normal y han sido explicados en unidades formativas anteriores, pero el alumno debe dominar su uso en los formularios.



El navegador Mozilla Firefox no lee y devuelve los colores en hexadecimal sino en RGB.

---

Los atributos son los siguientes:

### Básico

- Atributo id

Cuando a un formulario requiere ser identificado como un elemento único dentro del código de un programa HTML, como hemos visto anteriormente puede hacerse mediante el atributo "name", aunque es más elegante y correcto hacerlo mediante el atributo "id".

El atributo "id" tiene la misma funcionalidad que el "name", pero además puede ser utilizado como puntero de la hoja de estilos, o por JavaScript para manipular el formulario cuando se señala su id concreta.

Hay tres cosas importantes que deben cumplirse cuando se use este atributo:

- Debe contener al menos un carácter.
- No puede contener espacios.
- No diferencia entre mayúsculas y minúsculas.

Hay que tener mucho cuidado cuando se programa en php tras haber escrito código en HTML, porque el primero sí que diferencia entre mayúsculas y minúsculas, y es muy común acomodarse a un estilo de programación y olvidarse que los dos lenguajes son distintos en ese, y otros muchos sentidos.

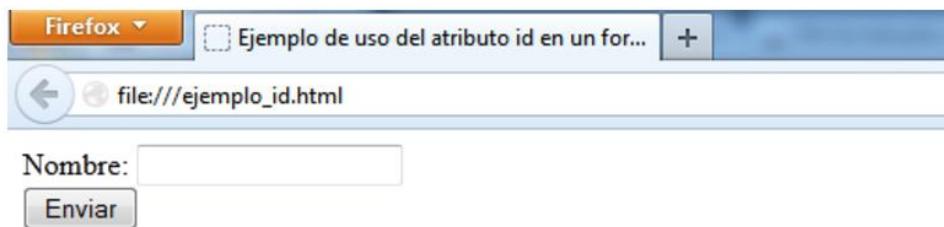
Aquí vemos el código de un ejemplo del atributo id.

```

<html>
  <head>
    <title>Ejemplo de uso del atributo id en un formulario.</title>
  </head>
  <body>
    <form action="formulario.php" method="post" id="form_envio_nombre">
      Nombre:
      <input type="text" name="nombre" value="" />
      <br/>
      <input type="submit" value="Enviar" />
    </form>
  </body>
</html>

```

Captura de pantalla del resultado que debe dar el código al ejecutarlo en un navegador web.



- Atributo class

Este atributo del formulario especifica uno o más nombres de clases asociados al mismo, con la intención de asociarle una o más clases CSS y teniendo sólo que programar esta para darle un aspecto general al diseño del formulario.

## UD1

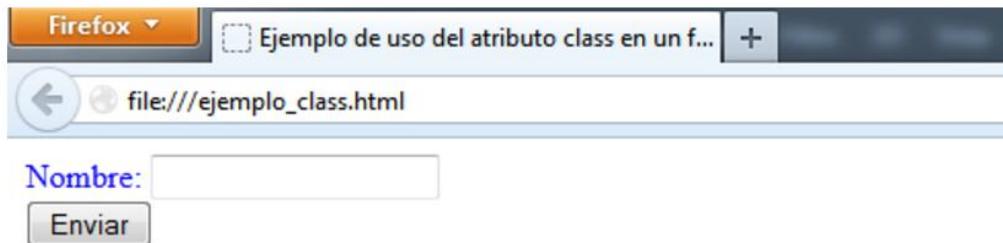
También este atributo se usa para apuntar a una hoja de estilos, y puede ser usado por JavaScript para cambiar elementos HTML usando unas determinadas clases que han sido programadas con esa intención. En el nombre de la clase se deben tener en cuenta los siguientes factores:

- Para especificar múltiples clases se deben separar por un espacio.
- El nombre de la clase debe empezar con una letra.
- Después del primer carácter, además de letras se pueden usar dígitos, guiones y guiones bajos.
- En HTML no se diferencian las mayúsculas y minúsculas.

Aquí vemos el código de un ejemplo del atributo class.

```
<html>
  <head>
    <title>Ejemplo de uso del atributo class en un formulario.</title>
    <style>
      sform.azul {color:blue;}
    </style>
  </head>
  <body>
    <form      action="formulario.php"      method="post"
      class="azul">
      Nombre:
      <input type="text" name="nombre" value="" />
      <br/>
      <input type="submit" value="Enviar" />
    </form>
  </body>
</html>
```

Captura de pantalla del resultado que debe dar el código al ejecutarlo en un navegador web.



- Atributo style

Si en lugar de usar el atributo "class", queremos definir el estilo del formulario directamente, el atributo que debemos usar es, sin duda alguna, el "style".

El atributo común "style" se caracteriza por aplicar, directamente, un estilo a cualquier elemento HTML, y en este caso se lo aplica directamente al formulario que queramos.

Este atributo estará por encima de cualquier otro estilo que se le haya aplicado al documento o al formulario anteriormente, como por ejemplo usando las etiquetas <style> en la cabecera, o en una clase CSS.

En este caso, los distintos estilos que se determinen mediante el uso de este atributo deben separarse por comas.

Aquí vemos el código de un ejemplo del atributo style.

```
<html>
<head>
    <title>Ejemplo de uso del atributo style en un formulario.</title>
</head>
<body>
    <form      action="formulario.php"      method="post"
style="color:blue">
        Nombre:
    </form>
</body>
```

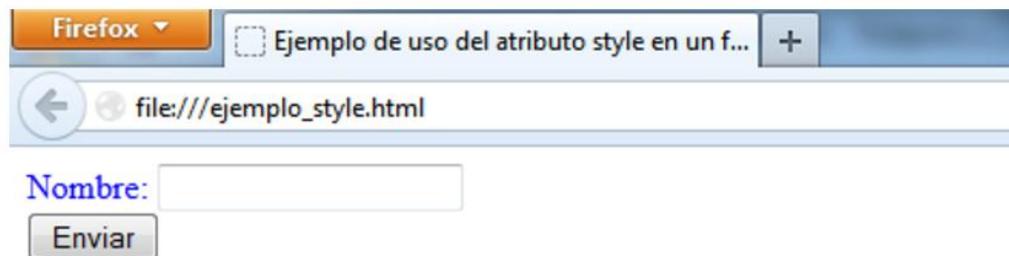
## UD1

```

<input type="text" name="nombre" value="" />
<br/>
<input type="submit" value="Enviar" />
</form>
</body>
</html>

```

Captura de pantalla del resultado que debe dar el código al ejecutarlo en un navegador web.



- Atributo title

Con este atributo tenemos la posibilidad de añadir más información específica al formulario mediante el uso de texto plano.

La información, dependiendo del tipo de formulario que se programe, puede ser simplemente una pista para el propio usuario cuando pase el ratón por encima, por ejemplo, o cualquier tipo de ayuda que pueda necesitar el usuario sobre el mismo.

Debe tenerse en cuenta que la información escrita, además de ser legible en texto plano normal, debe ser útil para el usuario, ya que esto no es un comentario específicamente puesto para futuros programadores, si no que es un elemento de cara al usuario final.

Aquí vemos el código de un ejemplo del atributo title.

```

<html>
<head>

```

```

<title>Ejemplo de uso del atributo title en un formulario.</title>

</head>

<body>

<form action="formulario.php" method="post"
      title="Formulario para enviar el nombre">

  Nombre:

  <input type="text" name="nombre" value="" />
  <br/>
  <input type="submit" value="Enviar" />
</form>

</body>

</html>

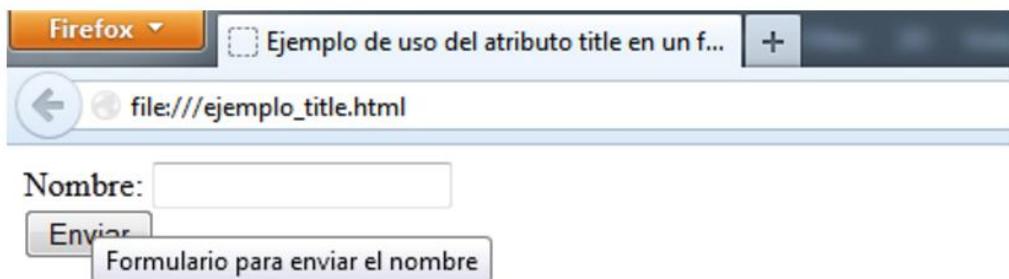
```



Este atributo no se ve su funcionamiento a simple vista si no se pasa el ratón por encima del formulario.

---

Captura de pantalla del resultado que debe dar el código al ejecutarlo en un navegador web.



# UD1

## Internacionalización

- Atributo lang

Cuando se programa un formulario no siempre está en el mismo idioma de la página en su conjunto. Es más, dependiendo del destino de dicho formulario, puede ser común que haya formularios de varios idiomas dentro de la misma página.

Cuando ocurren estas cosas, que puede dar problemas en la representación de los caracteres, o incluso en la autocorrección de algunos navegadores actuales dependientes del idioma, como ocurre con Google Chrome o Mozilla Firefox, se debe usar el atributo "lang". Este atributo permite indicar mediante el código HTML el lenguaje que va a ser utilizado en el formulario.



El código que define el idioma que se utiliza en español es el "es".

---

Aquí vemos el código de un ejemplo del atributo lang.

```
<html>
  <head>
    <title>Ejemplo de uso del atributo lang en un formulario.</title>
  </head>
  <body>
    <form action="formulario.php" method="post" lang="es">
      Nombre:
      <input type="text" name="nombre" value="" />
      <br/>
      <input type="submit" value="Enviar" />
    </form>
  </body>
</html>
```

```

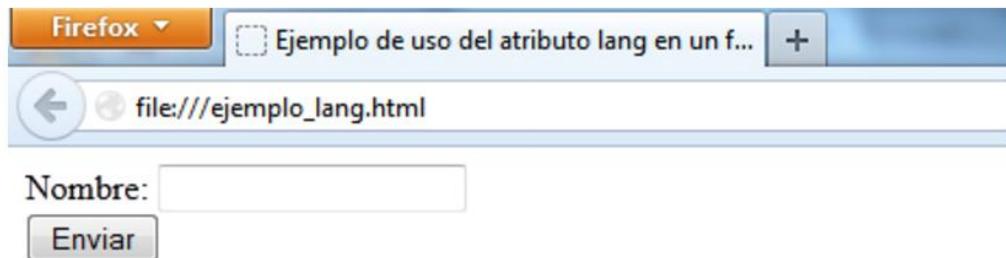
</form>

</body>

</html>

```

Captura de pantalla del resultado que debe dar el código al ejecutarlo en un navegador web.



A continuación tenemos los códigos de lenguaje más utilizados en el mundo, para usarlo como referencia al usar este atributo al programar en HTML.

LENGUAJE	CÓDIGO ISO
Albano	sq
Alemán	de
Árabe	ar
Armenio	hy
Bielorruso	be
Búlgaro	bg
Catalán	ca
Coreano	ko
Croata	hr
Checo	cs
Chino	zh
Chino (Simplificado)	zh-Hans
Chino (Tradicional)	zh-Hant
Danés	da
Eslovaco	sk
Esloveno	sl

# UD1

Español	es
Estonio	et
Euskera	eu
Finlandés	fi
Francés	fr
Gallego	gl
Griego	el
Hebreo	he, iw
Hindú	hi
Holandés	nl
Húngaro	hu
Indonesio	id, in
Inglés	en
Irlandés	ga
Islandés	is
Italiano	it
Japonés	ja
Lituano	lt
Macedonio	mk
Moldavo	mo
Mongol	mn
Nepalí	ne
Noruego	no
Polaco	pl
Portugués	pt
Rumano	ro
Ruso	ru
Serbio	sr
Serbo-Croata	sh
Sueco	sv
Tailandés	th
Turco	tr
Ucraniano	uk
Vietnamita	vi

- Atributo xml:lang:

Aunque el atributo "lang" es más que suficiente en la mayoría de situaciones que suele encontrarse un programador HTML, a veces este no es adecuado, especialmente cuando el contenido del formulario está directamente asociado con un documento XML.

Cuando el formulario se encuentra en esta situación es necesario usar el atributo "xml:lang" para mostrar el idioma de destino de dicho formulario.



El atributo "xml:lang" no es incompatible con el atributo "lang", de hecho puede ser posible que se encuentren conjuntamente en la misma etiqueta en la declaración de un formulario sin que por ello tengan ningún tipo de incompatibilidad.

---

Las variables que puede utilizar el atributo "xml:lang" son exactamente los mismos que los que usa el atributo "lang", por lo que la lista mostrada anteriormente puede usarse como referencia para el uso de este.

Aquí vemos el código de un ejemplo del atributo xml:lang.

```
<html>
<head>
    <title>Ejemplo de uso del atributo xml:lang en un formulario.</title>
</head>
<body>
    <form      action="formulario.php"      method="post"
xml:lang="es">
        Nombre:
        <input type="text" name="nombre" value="" />
        <br/>
```

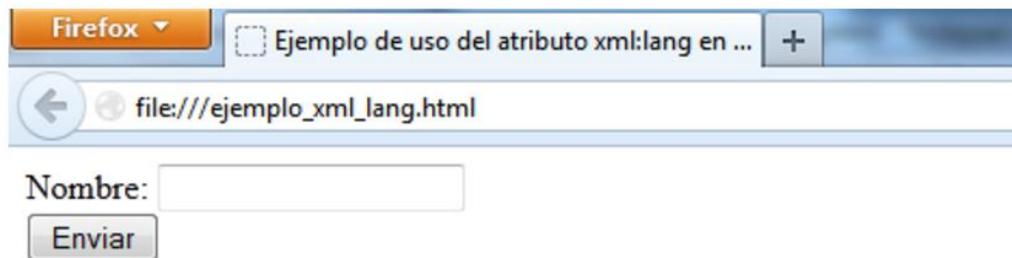
## UD1

```

<input type="submit" value="Enviar" />
</form>
</body>
</html>

```

Captura de pantalla del resultado que debe dar el código al ejecutarlo en un navegador web.



- Atributo dir:

El atributo "dir" se utiliza para especificar la dirección en la que irá el texto del formulario según el valor que se le haya asignado.

El uso de este atributo puede llegar a ser muy caótico y poco útil en la mayoría de los casos, pero es importante conocerlo y saber cuál es su utilidad para poder aplicarlo a formularios específicos que necesiten su aplicación.

En todo caso, en algunos idiomas es importante cambiar la dirección del texto por defecto, y utilizando junto con los atributos "lang" o "xml:lang" este atributo tiene, en algunos casos, una importancia notable.

Los valores que puede tomar el atributo "dir" son:

- ltr: Cuando el texto va de izquierda a derecha.
- rtl: Cuando el texto va de derecha a izquierda.
- auto: Cuando se deja que el navegador elija la dirección del texto.



El valor por defecto de este atributo, como es lógico, es el "ltr", que representa que el texto del formulario va de izquierda a derecha.

Aquí vemos el código de un ejemplo del atributo dir.

```
<html>
<head>
    <title>Ejemplo de uso del atributo dir en un formulario.</title>
</head>
<body>
    <form action="formulario.php" method="post" dir="rtl">
        Nombre:
        <input type="text" name="nombre" value="" />
        <br/>
        <input type="submit" value="Enviar" />
    </form>
</body>
</html>
```

Captura de pantalla del resultado que debe dar el código al ejecutarlo en un navegador web.



# UD1

## Eventos

- Atributo onblur

Los dos atributos que vamos a ver a continuación tienen una relación muy cercana, ya que realmente su función está ligada la una a la otra.

Además, en estos dos casos, ninguno de los dos son atributos de por sí de los formularios, sino más bien de los elementos del mismo, como son los botones, cajas de texto, áreas de texto, etc.

Sin embargo es necesario estudiarlos ahora, ya que al ser unos atributos generales de varios elementos, el alumno debe dominarlos antes de entrar a estudiar los mismos más adelante.

En concreto el atributo “onblur” entra en juego cuando el evento de nombre homónimo actúa, que es cuando el elemento del formulario al que se lo hemos asignado pierde el enfoque.

Aquí vemos el código de un ejemplo del atributo onblur.

```
<html>
  <head>
    <title>Ejemplo de uso del atributo onblur en un formulario.</title>
  </head>
  <body>
    <script type="text/javascript">
      function vaciar(control)
      {
        control.value='';
      }
      function verificarFormulario(control)
      {
        if (control.value=='')
      }
    </script>
  </body>
</html>
```

## UF1304: Elaboración de plantillas y formularios

```
        alert('Debe llenar el formulario');

    }

</script>

<form>

    <input type="text" id="nombre" onFocus="vaciar(this)" onBlur="verificarFormulario(this)" value="nombre">

    <br/>

    <input type="text" id="edad" onFocus="vaciar(this)" onBlur="verificarFormulario(this)" value="edad">

    <br/>

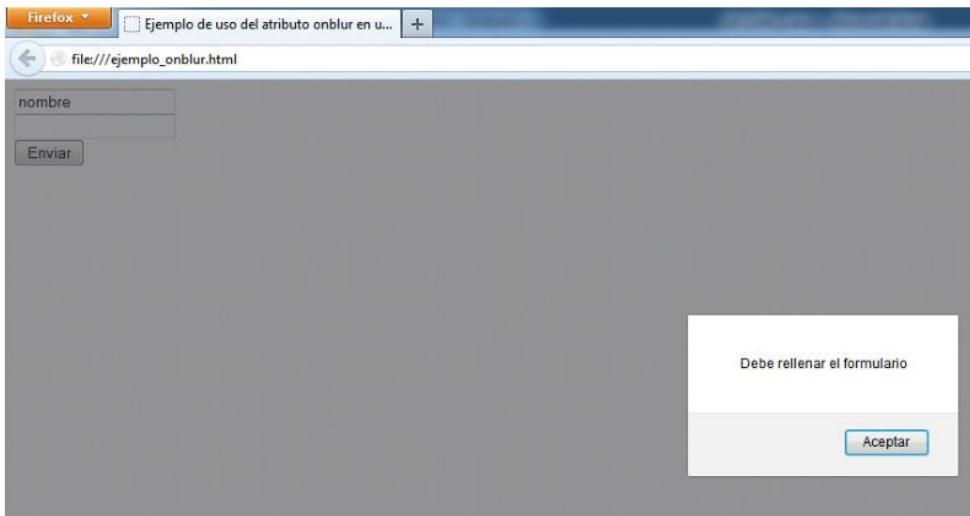
    <input type="button" value="Enviar">

</form>

</body>

</html>
```

Captura de pantalla del resultado que debe dar el código al ejecutarlo en un navegador web.



## UD1

- Atributo focus:

Al contrario que el atributo “onblur”, el atributo “onfocus” actúa cuando el formulario obtiene el enfoque.

Ambos atributos son antagónicos, y aunque pueden ser programados independientemente el uno del otro, cobran más sentido cuando se ven juntos.



Importante

En el siguiente código, que es el mismo que hemos podido ver en el ejemplo anterior, cuando se pulsa en un lugar distinto al los eventos del formulario con los atributos y pierden el enfoque, si están vacíos sale un mensaje avisando. Cuando esos elementos obtienen el foco al pulsar sobre ellos, gracias al atributo “onfocus” se vacían.

---

Aquí vemos el código de un ejemplo del atributo onfocus.

```
<html>
<head>
    <title>Ejemplo de uso del atributo onfocus en un formulario.</title>
</head>
<body>
    <script type="text/javascript">
        function vaciar(control)
        {
            control.value='';
        }
        function verificarFormulario(control)
    </script>

```

```

{
    if (control.value=='')
        alert('Debe llenar el formulario');

}
</script>

<form>

<input type="text" id="nombre" onFocus="vaciar(this)" onBlur="verificarFormulario(this)" value="nombre">

<br/>

<input type="text" id="edad" onFocus="vaciar(this)" onBlur="verificarFormulario(this)" value="edad">

<br/>

<input type="button" value="Enviar">

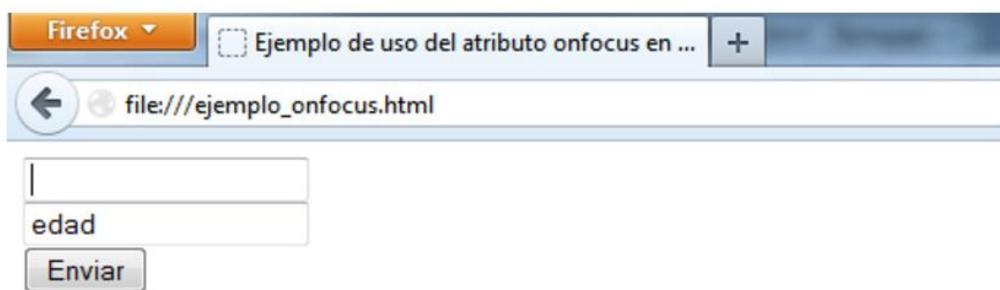
</form>

</body>

</html>

```

Captura de pantalla del resultado que debe dar el código al ejecutarlo en un navegador web.



- Atributo onchange:

Este atributo de HTML sólo se puede aplicar a ciertos elementos de un formulario, no al formulario en sí.

## UD1

El funcionamiento de este atributo es actuar como un evento que se ejecuta cuando un elemento como un cuadro de texto pierde el enfoque, y además su valor ha sido cambiado.

Cuando se ejecuta en una casilla de texto no se activa hasta que se pasa a otra casilla.

Aquí vemos el código de un ejemplo del atributo onchange.

```
<html>
<head>
    <title>Ejemplo de uso del atributo onchange en un formulario.</title>
</head>
<body>
    <script type="text/javascript">
        function cambioFormulario(control)
        {
            if (control)
                alert('Ha cambiado un elemento del formulario');
        }
    </script>
    <form>
        <input type="text" id="nombre" onChange="cambioFormulario(this)" value="nombre">
        <br/>
        <input type="text" id="edad" onChange="cambioFormulario(this)" value="edad">
        <br/>
        <input type="button" value="Enviar">
    </form>
</html>
```

```

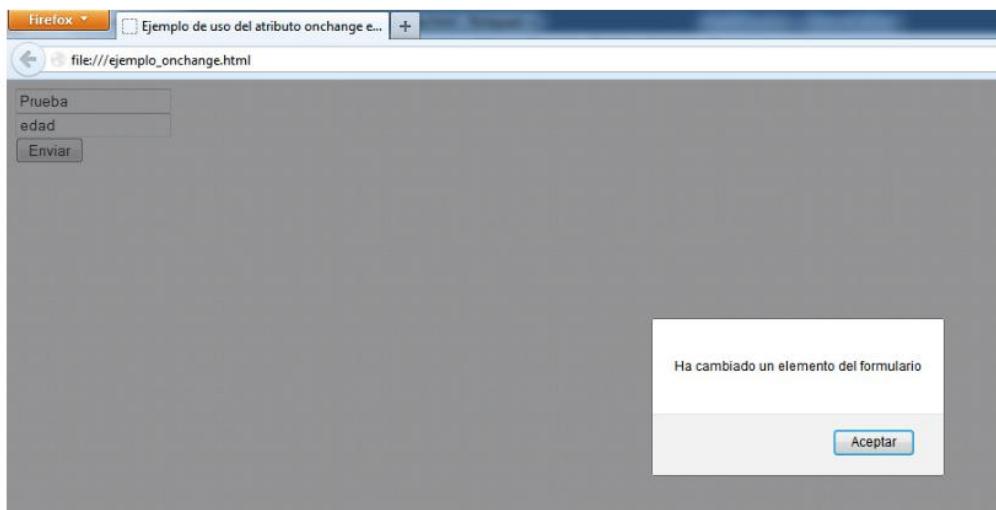
</form>

</body>

</html>

```

Captura de pantalla del resultado que debe dar el código al ejecutarlo en un navegador web.



- Atributo onclick:

Este atributo y el siguiente están relacionados entre sí, ya que actúan de la misma forma, con la diferencia de que depende del número de pulsaciones que se le dé al ratón para que actúen.

El atributo “onclick” actúa como un evento que se lanza cuando el usuario hace click con el ratón en el formulario o el elemento del formulario al que se le ha asociado el atributo.

Y por supuesto, esto activará la función de script asociada.

Aquí vemos el código de un ejemplo del atributo onclick.

```

<html>

<head>

    <title>Ejemplo de uso del atributo onclick en un formulario.</title>

```

## UD1

```

<script language=JavaScript>

var numerodeclicks = 0;

function miboton_onclick()

{

    numerodeclicks++;

    window.document.formulario.miboton.value = 'Bo-
ton pulsado ' + numerodeclicks + ' veces';

}

</script>

</head>

<body>

<form name=formulario>

<input type='button' name='miboton' value='Boton pul-
sado 0 veces' onclick="miboton_onclick()">

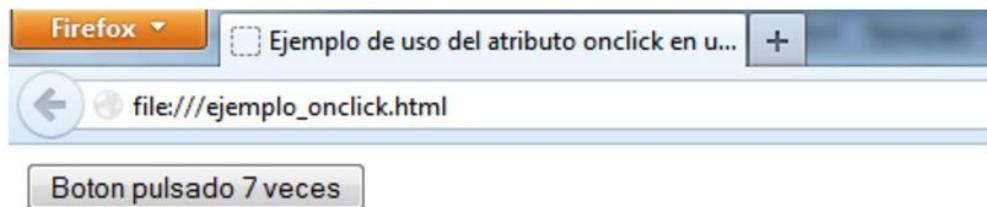
</form>

</body>

</html>

```

Captura de pantalla del resultado que debe dar el código al ejecutarlo en un navegador web.



- Atributo ondblclick:

El atributo "ondblclick" actúa de forma similar al atributo "onclick", como un evento que se lanza cuando el usuario hace doble click con el ratón, en este caso, en el formulario o el elemento del formulario al que se le ha asociado el atributo. Esto activará la función de script asociada que habremos programado anteriormente.

Aquí vemos el código de un ejemplo del atributo onclick.

```
<html>
<head>

    <title>Ejemplo de uso del atributo ondblclick en un formulario.</title>

    <script language=JavaScript>

        var numerodeclicks = 0;

        function miboton_ondblclick()

        {

            numerodeclicks++;

            window.document.formulario.miboton.value = 'Doble click hecho ' + numerodeclicks + ' veces';

        }

    </script>

</head>

<body>

    <form name=formulario>

        <input type='button' name='miboton' value='Doble click hecho 0 veces' ondblclick="miboton_ondblclick()">

    </form>

</body>

</html>
```

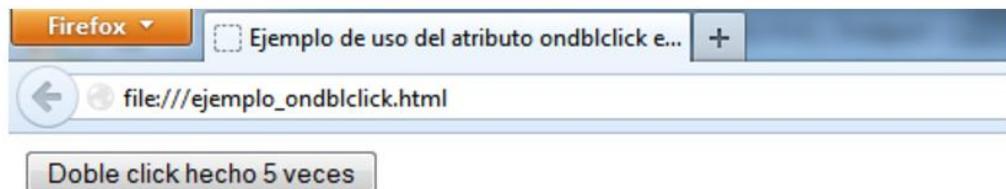
## UD1



En este caso para que se active la función del botón, obviamente habrá que hacer doble click sobre él.

---

Captura de pantalla del resultado que debe dar el código al ejecutarlo en un navegador web.



- Atributo onkeydown:

Tanto este atributo como los dos siguientes están relacionados entre sí.

En concreto, tanto el atributo “onkeydown” como los atributos “onkeypress” y “onkeyup” se aplican a los elementos del formulario a los que se asocie.

El atributo “onkeydown” hace que, el elemento del formulario que lo tiene asociado, active el evento homónimo cuando el usuario pulsa una tecla en dicho elemento.

Aquí vemos el código de un ejemplo del atributo onkeydown.

```
<html>
<head>
  <title>Ejemplo de uso del atributo onkeydown en un formulario.</title>
  <script language=JavaScript>
```

```

        function MostrarEvento(nombredelevento)

        {
            var mensaje = window.document.formulario.textarea2.value;

            mensaje = mensaje + 'Se ha usado el evento: ' +
nombredelevento;

            window.document.formulario.textarea2.value      =
mensaje;

        }

    </script>

</head>

<body>

<form name=formulario>

    <textarea rows=10 cols=30 name=textareal onkeydown="MostrarEvento('onkeydown\n');">

```

</textarea>

```

    <textarea rows=10 cols=30 NAME=textarea2>

```

</textarea>

```

    <br/>

```

```

    <br/>

```

```

    <input type="button" value="Limpiar la zona de texto
derecha"

```

```

        name=button1 onclick="window.document.formulario.textarea2.value=''">

```

```

</form>

```

```

</body>

```

```

</html>

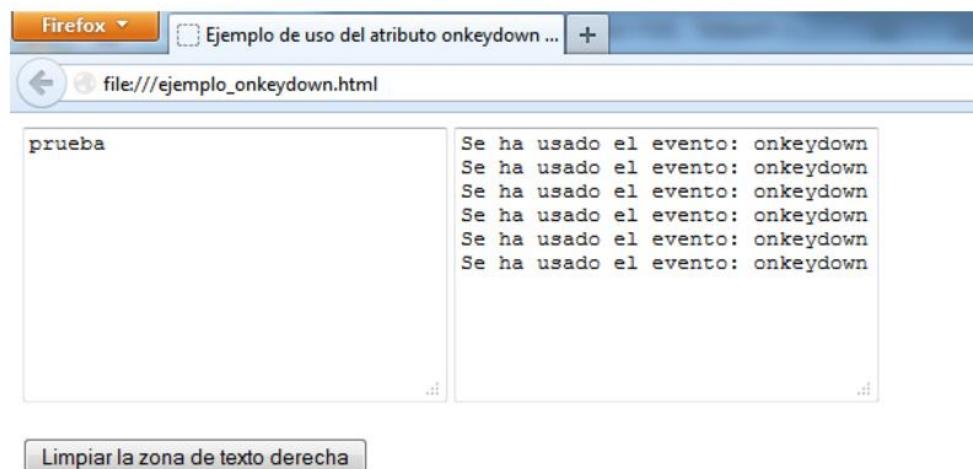
```

## UD1



Tanto este código como los códigos de las dos siguientes variables sirven para ver, visualmente, cuando se activan los eventos asociados a los atributos explicados.

Captura de pantalla del resultado que debe dar el código al ejecutarlo en un navegador web.



- Atributo onkeypress:

El atributo “onkeypress” también se aplica a los elementos del formulario a los que se asocia, no al formulario en sí.

El atributo “onkeypress” hace que, el elemento del formulario que lo tiene asociado, active el evento homónimo cuando el usuario mantiene pulsada una tecla en dicho elemento.

Aquí vemos el código de un ejemplo del atributo onkeypress.

```
<html>
<head>
  <title>Ejemplo de uso del atributo onkeypress en un formulario.</title>
```

```
<script language=JavaScript>

function MostrarEvento(nombredelevento)

{
    var mensaje = window.document.formulario.textarea2.value;

    mensaje = mensaje + 'Se ha usado el evento: ' +
nombredelevento;

    window.document.formulario.textarea2.value      =
mensaje;

}

</script>

</head>

<body>

<form name=formulario>

<textarea rows=10 cols=30 name=textareal onkeypress="
MostrarEvento('onkeypress\n');" >

</textarea>

<textarea rows=10 cols=30 NAME=textarea2>

</textarea>

<br/>

<br/>

<input type="button" value="Limpiar la zona de texto
derecha"

name=button1 onclick="window.document.formulario.tex-
tarea2.value=''">

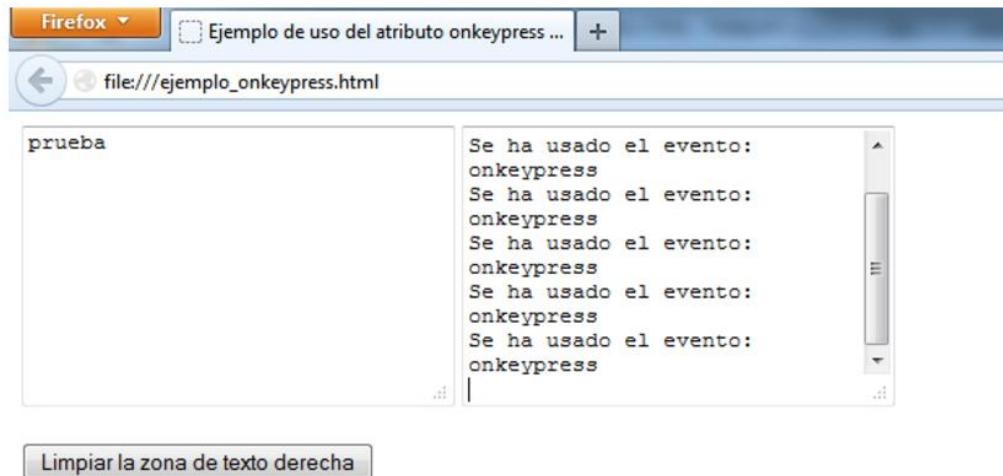
</form>

</body>

</html>
```

## UD1

Captura de pantalla del resultado que debe dar el código al ejecutarlo en un navegador web.



- Atributo onkeyup:

El atributo “onkeyup”, como los dos anteriores, se aplica a los elementos del formulario a los que se añade, no al formulario en sí.

El atributo “onkeyup” hace que, el elemento del formulario que lo tiene asociado, active el evento con el mismo nombre cuando el usuario deja de pulsar una tecla en dicho elemento.

Aquí vemos el código de un ejemplo del atributo onkeyup.

```
<html>
<head>
    <title>Ejemplo de uso del atributo onkeyup en un formulario.</title>
    <script language=JavaScript>
        function MostrarEvento(nombredelevento)
        {
            var mensaje = window.document.formulario.textarea2.value;
```

## UF1304: Elaboración de plantillas y formularios

```

        mensaje = mensaje + 'Se ha usado el evento: ' +
nombredeevento;

        window.document.formulario.textarea2.value      =
mensaje;

    }

</script>

</head>

<body>

<form name=formulario>

<textarea rows=10 cols=30 name=textareal onkeyup="MostrarEvento('onkeyup\n');">

</textarea>

<textarea rows=10 cols=30 NAME=textarea2>

</textarea>

<br/>

<br/>

<input type="button" value="Limpiar la zona de texto
derecha"

name=button1 onclick="window.document.formulario.tex-
tarea2.value=''">

</form>

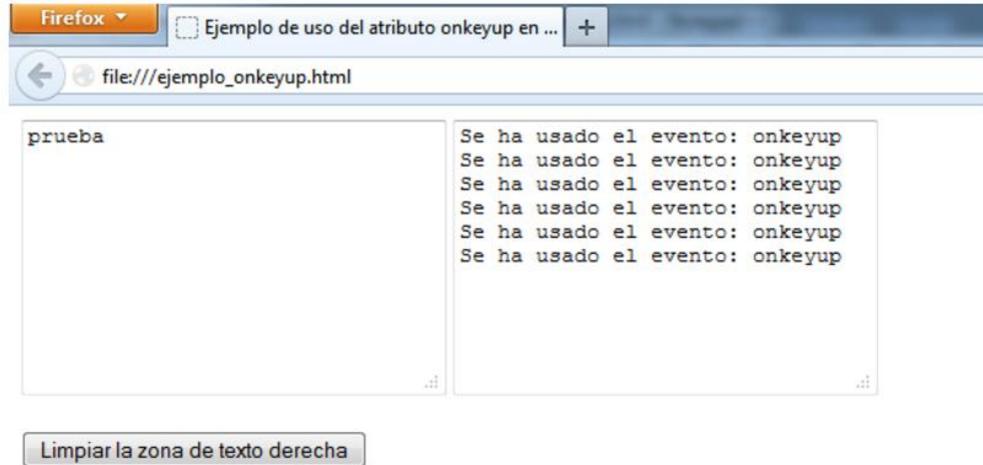
</body>

</html>

```

Captura de pantalla del resultado que debe dar el código al ejecutarlo en un navegador web.

## UD1



- Atributo onmousedown:

Tanto este atributo como los dos siguientes están relacionados entre sí. En concreto, tanto el atributo "onmousedown" como los atributos "onmousemove" y "onmouseup" se aplican a los elementos del formulario a los que se asocie.

El atributo "onmousedown" hace que, el elemento del formulario que lo tiene asociado, active el evento homónimo cuando el usuario pulsa botón del ratón cuando el puntero está en dicho elemento.

Aquí vemos el código de un ejemplo del atributo onmousedown.

```
<html>
<head>
    <title>Ejemplo de uso del atributo onMouseDown en un formulario.</title>
    <script language=JavaScript>
        function boton_onmouseup ()
        {
            document.formulario.boton.value = "El raton no
esta pulsado"
        }
        function boton_onmousedown ()
    </script>
</head>
<body>
    <form name=formulario>
        <input type=button value="Pulsar" name=boton>
    </form>
</body>

```

```

    {
        document.formulario.boton.value = "El raton esta
pulsado"
    }

    function boton_onmousemove()
    {
        document.formulario.boton.value = "El puntero
esta sobre el boton"
    }

</script>

</head>

<body>

<form name=formulario>

<input type='button' name='boton' value='El raton no
esta pulsado ni sobre el boton'
onmouseup="boton_onmouseup()"
onmousedown="boton_onmousedown()"
onmousemove="boton_onmousemove()">

</form>

</body>

</html>

```

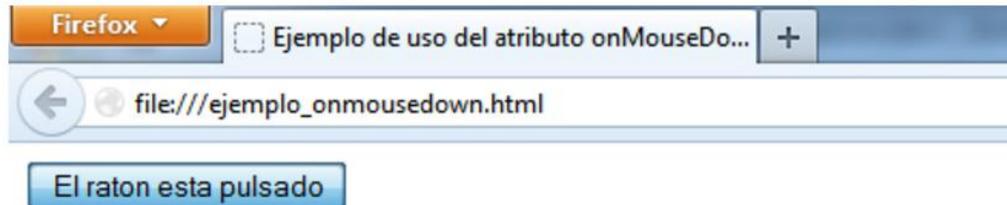


Dependiendo si el ratón se mueve por encima del botón, si se pulsa o si se deja de pulsar el botón, se activarán los respectivos atributos, "onmousemove", "onmousedown" o "onmouseup".

---

## UD1

Captura de pantalla del resultado que debe dar el código al ejecutarlo en un navegador web.



- Atributo onmousemove:

El atributo “onmousemove” también se aplica a los elementos del formulario a los que se asocia, no al formulario en sí. El atributo “onmousemove” hace que, el elemento del formulario que lo tiene asociado, active el evento homónimo cuando el usuario mueve el puntero del ratón por encima de dicho elemento.

Aquí vemos el código de un ejemplo del atributo onmousemove.

```
<html>
<head>
    <title>Ejemplo de uso del atributo onmousemove en un formulario.</title>
    <script language=JavaScript>
        function boton_onmouseup()
        {
            document.formulario.boton.value = "El raton no
esta pulsado"
        }
        function boton_onmousedown()
        {
            document.formulario.boton.value = "El raton esta
pulsado"
        }
    </script>
</head>
<body>
    <form name=formulario>
        <input type=button value="El raton no
esta pulsado" name=boton>
    </form>
</body>

```

```

function boton_onmousemove()
{
    document.formulario.boton.value = "El puntero
esta sobre el botón"

}

</script>

</head>

<body>

<form name=formulario>

<input type='button' name='boton' value='El ratón no
esta pulsado ni sobre el botón'

onmouseup="boton_onmouseup()"
onmousedown="boton_onmousedown()"
onmousemove="boton_onmousemove()"

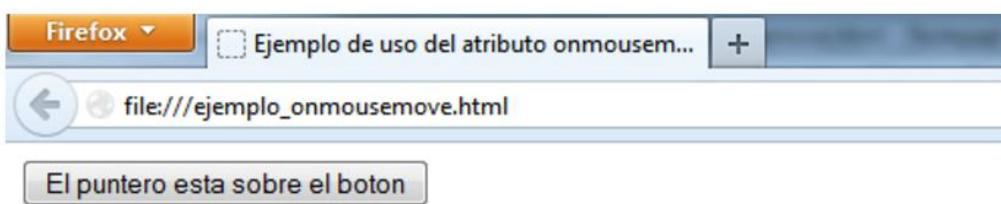
</form>

</body>

</html>

```

Captura de pantalla del resultado que debe dar el código al ejecutarlo en un navegador web.



- Atributo onmouseup:

El atributo “onmouseup”, como los dos anteriores, se aplica a los elementos del formulario a los que se añade, no al formulario en sí.

## UD1

El atributo “onmouseup” hace que, el elemento del formulario que lo tiene asociado, active el evento con el mismo nombre cuando el usuario deja de pulsar con el ratón en dicho elemento.

Aquí vemos el código de un ejemplo del atributo onmouseup.

```

<html>
  <head>
    <title>Ejemplo de uso del atributo onmouseup en un formulario.</title>
    <script language=JavaScript>
      function boton_onmouseup()
      {
        document.formulario.boton.value = "El raton no
esta pulsado"
      }

      function boton_onmousedown()
      {
        document.formulario.boton.value = "El raton esta
pulsado"
      }

      function boton_onmousemove()
      {
        document.formulario.boton.value = "El puntero
esta sobre el boton"
      }
    </script>
  </head>
  <body>
    <form name=formulario>

```

```

<input type='button' name='boton' value='El raton no
esta pulsado ni sobre el boton'

onmouseup="boton_onmouseup()"
onmousedown="boton_onmousedown()"
onmousemove="boton_onmousemove()">

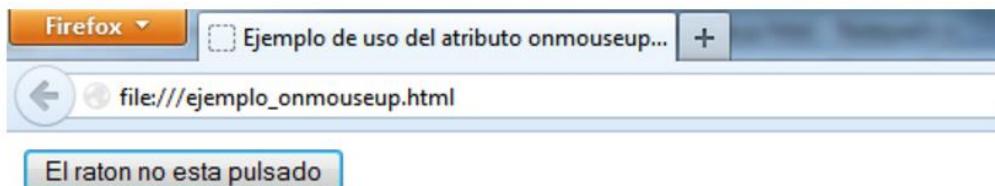
</form>

</body>

</html>

```

Captura de pantalla del resultado que debe dar el código al ejecutarlo en un navegador web.



- Atributo onmouseout:

Al igual que ocurre anteriormente con otros atributos asociados a los elementos del formulario o al mismo formulario, los dos eventos que vamos a ver a continuación están muy relacionados entre sí.

En concreto, el atributo onmouseout se utiliza cuando queremos controlar lo que ocurre cuando el puntero del ratón abandona un formulario, o un elemento del formulario. Producíendose el efecto que programemos.

Aquí vemos el código de un ejemplo del atributo onmouseout.

```

<html>

<head>

    <title>Ejemplo de uso del atributo onmouseout en un
    formulario.</title>

</head>

```

## UD1

```

<body>

    <script type="text/javascript">
        function colorear(color)
        {
            document.bgColor=color;
        }
    </script>

    <form>

        <input type="text" id="nombre" onMouseOver="colorear('
#ff0000')" onMouseOut="colorear('#ff00ff')">

        <br/>

        <input type="button" value="Enviar">

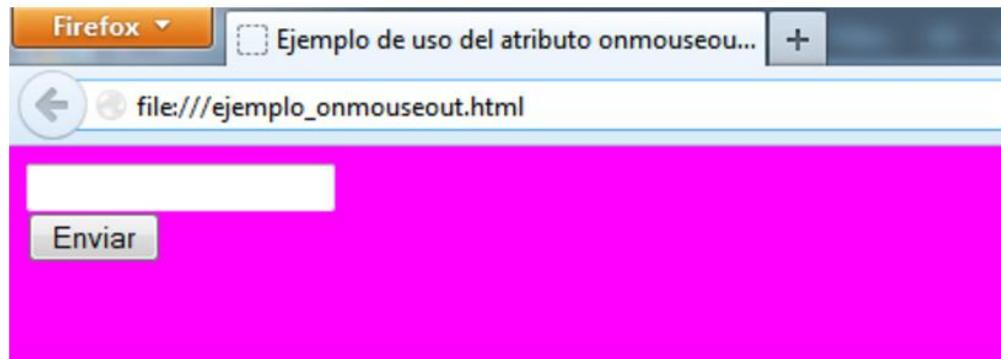
    </form>

</body>

</html>

```

Captura de pantalla del resultado que debe dar el código al ejecutarlo en un navegador web.



- Atributo onmouseover:

El atributo onmouseover se utiliza cuando queremos controlar lo que ocurre cuando el puntero del ratón pasa sobre un formulario, o un elemento del formulario. Producéndose el efecto que programemos.



En este caso, cuando pasamos el ratón por encima del cuadro de texto del formulario, este se pone de color rojo, y cuando el puntero abandona el cuadro de texto, el formulario se pone de color morado.

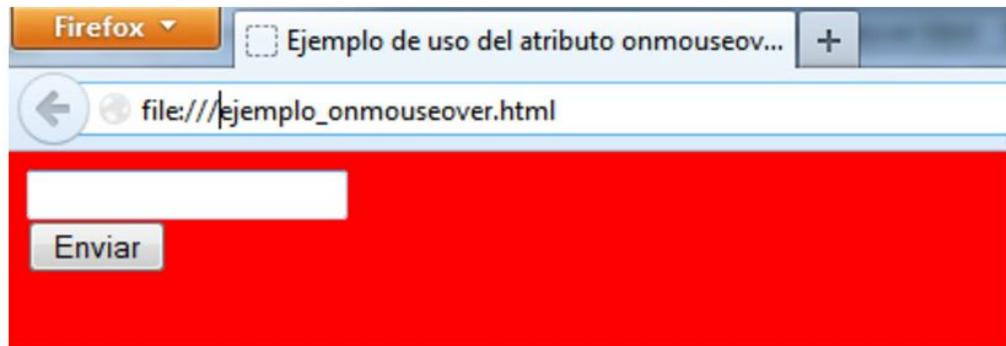
---

Aquí vemos el código de un ejemplo del atributo onmouseover.

```
<html>
<head>
    <title>Ejemplo de uso del atributo onmouseout en un formulario.</title>
</head>
<body>
    <script type="text/javascript">
        function colorear(color)
        {
            document.bgColor=color;
        }
    </script>
    <form>
        <input type="text" id="nombre" onMouseOver="colorear('
#ff0000')" onMouseOut="colorear('#ff00ff')">
        <br/>
        <input type="button" value="Enviar">
    </form>
</body>
</html>
```

## UD1

Captura de pantalla del resultado que debe dar el código al ejecutarlo en un navegador web.



- Atributo onreset:

Este atributo sólo se puede asociar a los formularios, y su función es asociar el botón de reset del mismo con una función de JavaScript.

Por lo tanto, en la declaración de la etiqueta del formulario, si se asigna este atributo a una función previamente declarada, al pulsar sobre el botón de reset esta sería activada.

El efecto que se active se ejecutará además del borrado de la información introducida en el formulario, por supuesto.

Aquí vemos el código de un ejemplo del atributo onreset.

```
<html>
  <head>
    <title>Ejemplo de uso del atributo onreset en un formulario.</title>
  </head>
  <body>
    <script type="text/javascript">
      function colorear(color)
      {
        document.bgColor=color;
      }
    </script>
  </body>
</html>
```

```

</script>

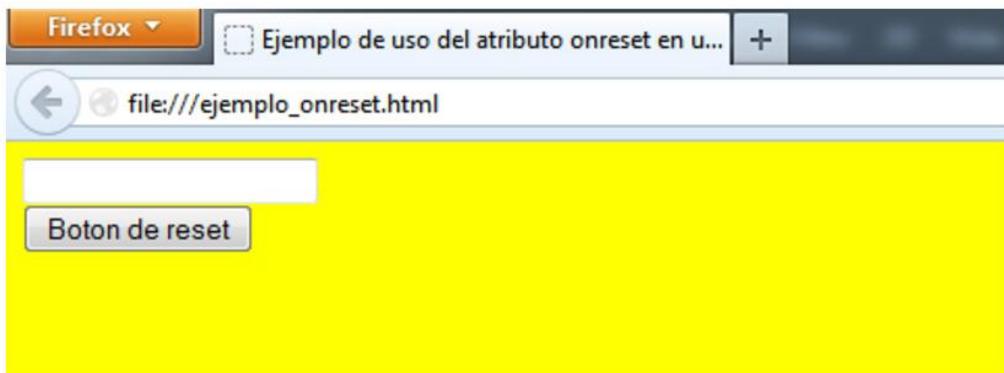
<form onReset="colorear('#ff0000')">
  <input type="text" id="nombre" >
  <br/>
  <input type="reset" value="Boton de reset">
</form>

</body>

</html>

```

Captura de pantalla del resultado que debe dar el código al ejecutarlo en un navegador web.



- Atributo onselect:

Este atributo sólo se puede asociar a algunos elementos de los formularios, y su función es asociar cuando texto de ese elemento es seleccionado con una función de JavaScript.

Por lo tanto, en la declaración de la etiqueta del elemento, si se asigna este atributo a una función previamente declarada, al seleccionar texto de este elemento, como por ejemplo un cuadro de texto, esta sería activada.

Aquí vemos el código de un ejemplo del atributo onselect.

```

<html>
  <head>

```

## UD1

```
<title>Ejemplo de uso del atributo onselect en un formulario.</title>

</head>

<body>

    <script type="text/javascript">

        function colorear(color)

        {

            document.bgColor=color;

        }

    </script>

    <form>

        <input type="text" id="nombre" onSelect="colorear('#00ff00')"

        <br/>

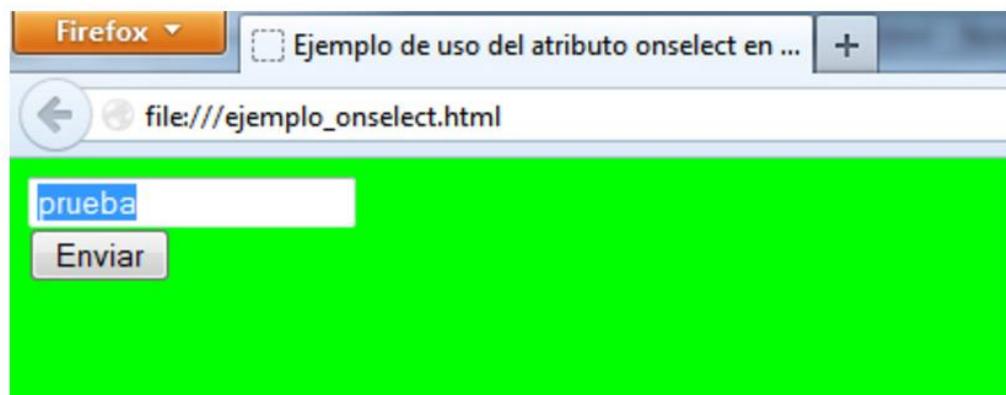
        <input type="button" value="Enviar">

    </form>

</body>

</html>
```

Captura de pantalla del resultado que debe dar el código al ejecutarlo en un navegador web.



- Atributo onsubmit:

Este atributo sólo se puede asociar a los formularios, y su función es asociar el botón de enviar del mismo con una función de JavaScript.

Por lo tanto, en la declaración de la etiqueta del formulario, si se asigna este atributo a una función previamente declarada, al pulsar sobre el botón de enviar esta sería activada.

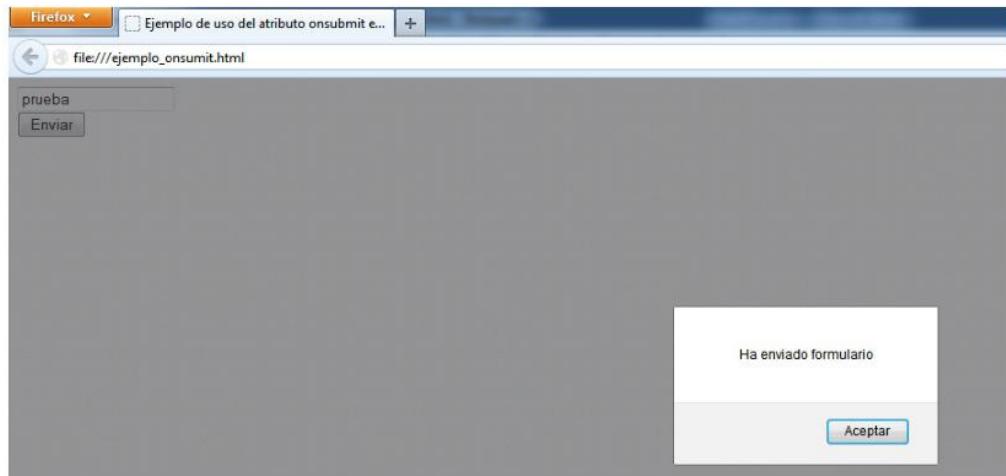
El efecto que se active se ejecutará además del enviado de la información introducida en el formulario al servidor, por supuesto.

Aquí vemos el código de un ejemplo del atributo onsubmit.

```
<html>
<head>
    <title>Ejemplo de uso del atributo onsubmit en un formulario.</title>
</head>
<body>
    <script type="text/javascript">
        function enviarFormulario()
        {
            alert('Ha enviado formulario');
        }
    </script>
    <form onSubmit="enviarFormulario()">
        <input type="text" id="nombre" >
        <br/>
        <input type="submit" value="Enviar">
    </form>
</body>
</html>
```

## UD1

Captura de pantalla del resultado que debe dar el código al ejecutarlo en un navegador web.



### Foco

- Atributo accesskey:

Con este atributo se puede asignar una tecla de acceso rápido a un formulario o a un elemento de un formulario.

El acceso a estos elementos varía dependiendo del navegador que se utilice. En concreto, las formas de acceso según el navegador son:

- Internet Explorer: Hay que pulsar ALT más la tecla de acceso.
- Opera: Hay que pulsar ALT, más ESCAPE, más la tecla de acceso.
- El resto de navegadores: Hay que pulsar ALT, más SHIFT, más la tecla de acceso.



A pesar de estas combinaciones la mayoría de los navegadores permiten modificar su configuración para cambiar la combinación de teclas que es necesaria para usar este atributo junto con una tecla de acceso.

Al declarar el atributo debe igualarse al carácter que se quiera que sea el acceso rápido.

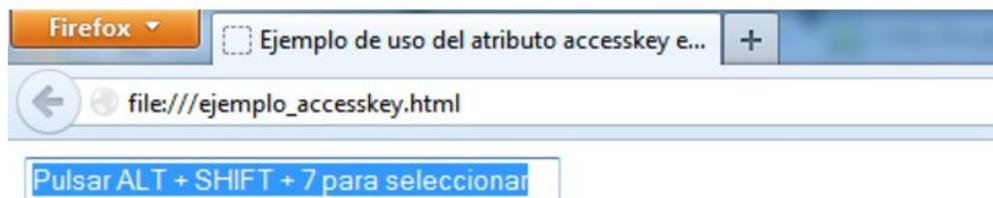
Aquí vemos el código de un ejemplo del atributo accesskey.

```
<html>
<head>
    <title>Ejemplo de uso del atributo accesskey en un formulario.</title>
</head>
<body>
    <form name=formulario>
        <input type="text" size="40" value="Pulsar ALT + SHIFT
+ 7 para seleccionar" accesskey="7">
    </form>
</body>
</html>
```



Tras seleccionar el elemento del formulario se puede editar o no, pero eso ya depende del usuario, no tiene nada que ver con el uso de este atributo.

Captura de pantalla del resultado que debe dar el código al ejecutarlo en un navegador web.



## UD1

- Atributo tabindex:

Con este atributo se determina el orden de los elementos del formulario cuando se pasa de uno a otro usando la tabulación.

Normalmente el orden está determinado por la declaración de los elementos en sí, pero gracias a este elemento el programador tiene flexibilidad para cambiarlo, de forma que si considera que los campos deben ser usados en un determinado orden puede cambiar el orden de acceso a ellos.

Este atributo también no es usado en el formulario, ya que no es un objeto HTML que pueda ser seleccionado.

Aquí vemos el código de un ejemplo del atributo tabindex.

```
<html>
  <head>
    <title>Ejemplo de uso del atributo tabindex en un formulario.</title>
  </head>
  <body>
    <form>
      Usuario:
      <input type="text" name="usuario" value="" tabindex="4"/>
      <br/>
      Password:
      <input type="password" name="contrasena" value="" tabindex="5"/>
      <br/>
      Repetir Password:
      <input type="password" name="confirmacion_contrasena" value="" tabindex="2"/>
    </form>
  </body>
</html>
```

```

<br/>

Subir avatar de usuario:

<input type="file" name="avatar" tabindex="3"/>

<br/>

<input type="submit" value="Enviar" tabindex="2"/>

</form>

</body>

</html>

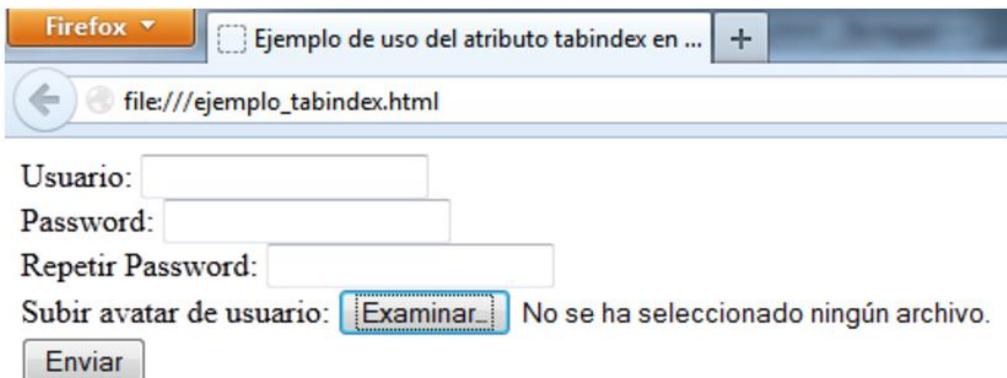
```



Si se van seleccionando los elementos con el tabulador se comprobará que el orden está cambiado.

---

Captura de pantalla del resultado que debe dar el código al ejecutarlo en un navegador web.



## UD1

### 1.2.1. Descripción y definición de los elementos de un formulario

Una vez seleccionados los atributos específicos del formulario dentro de la etiqueta <form>, llega el momento de completar el mismo poniendo los distintos elementos que, como vimos en las características de los formularios, son vitales para seleccionar la información que se quiere que el usuario envíe mediante el uso del mismo.

Estos elementos están hechos todos basados en las distintas casillas que se fueron implementando en los formularios reales, como son:

- Los cuadros de texto.
- Los cuadros ocultos usados para contraseñas.
- Las casillas de verificación.
- Los botones de opciones.
- Los botones de envío de información.
- Los botones de reseteo de la información.
- Las herramientas para adjuntar archivos.
- Los botones usando imágenes.
- Los campos ocultos al usuario.
- Los botones para otras funciones específicas.
- Las áreas de texto donde se pueden introducir muchos más caracteres.
- Las listas de selección.
- Las etiquetas para agrupar elementos del formulario.
- Las herramientas para agrupar los elementos del formulario.
- Los elementos que nos sirven para nombrar o etiquetar esos agrupamientos creados.

- Y sobre todo, los distintos elementos que hacen que un formulario sea precisamente lo que necesitamos en cualquier diseño funcional que necesite el cliente, de cara al usuario final que vaya a usar esa aplicación.



Cuando se programa cualquier elemento de un formulario, se debe ser muy consciente del objetivo final y de su uso, y de utilizar los elementos adecuados en cada momento para ese formulario. Por ejemplo, se podría usar un simple cuadro de texto de tamaño normal para introducir una edad o una fecha, pero si en el primer caso usamos un cuadro de texto más estrecho, y en el segundo un cuadro tipo "date", el formulario será mucho más correcto y estará mucho mejor programado.

Captura de pantalla del formulario de ingreso cuando se intenta hacer una nueva cuenta en gmail. Es un ejemplo perfecto del uso de distintos tipos de cajas de texto y de tamaños adecuados para cada información que se necesita. Además de otros elementos de formulario como son las listas de selección.

**Crea tu cuenta de Google**

Solo necesitas una cuenta  
Accede a todos los servicios de Google con solo un nombre de usuario y una contraseña.

g M G C Y D A S

Personaliza Google a tu gusto  
Configura tu perfil y tus preferencias como más te guste.

Tu información siempre disponible

Nombre  Apellidos   
Nombre de usuario  @gmail.com  
Contraseña   
Confirma tu contraseña   
Fecha de nacimiento  Dia  Mes  Año   
Sexo  Selecciona tu sexo   
Teléfono móvil  ES +34  
Tu dirección de correo electrónico actual

Teniendo presente estos detalles, el formulario no solo tendrá mucho mejor diseño, sino que también será mucho más funcional.

## UD1

Por enumerar una serie de casos claros, un simple cuadro de texto puede ser utilizado para:

- Introducir texto plano sin más.
- Introducir una fecha.
- Introducir un email.
- Introducir una URL.
- Introducir un teléfono.

O para cualquier otro dato que se diferencie de los demás en algo importante, o que simplemente esté acotado a un número de valores bajo.

Un ejemplo muy típico es el de los cuadros de texto para pedir la edad. Es muy fácil enumerar los controles que debería tener un cuadro de texto que pida este dato:

- Debe ser corto.
- Debe aceptar sólo números.
- El valor debe ser mayor que cero.
- El número no puede tener más de tres cifras.

A continuación vemos el ejemplo de cómo se puede usar un cuadro de texto para múltiples funciones dentro de un mismo formulario, en el registro de la web para aprender idiomas, duolingo.

Por un lado tenemos un cuadro de texto normal para introducir el nombre completo pero que no es necesario llenar, por lo que completarlo o no es opcional. Además de tener un tamaño considerable, para que entre en el cuadro un nombre muy largo, deberá programarse que es opcional.

A continuación tenemos un cuadro de texto para el correo. Este se caracterizará, además de por qué tiene que ser obligatorio, para mandar el correo de confirmación del registro, debe tener el formato adecuado para los emails, como es, por ejemplo, que el nombre introducido tenga una arroba en medio de los caracteres.

Acto seguido está un cuadro de texto para el usuario. Este cuadro de texto en principio debería ser más parecido al del nombre, sin embargo, nos encontra-

mos con que es obligatorio, y que, además, es mucho más corto que el del nombre, ya que los identificadores de usuario siempre suelen tener un número de caracteres acotado.

Y por último nos encontramos con el cuadro de texto de la contraseña, un lugar donde, además del tipo de elemento, que oculta los caracteres escritos en la pantalla, los datos deberán ser enviados lo más cifrados posibles para evitar la posibilidad de problemas de seguridad.

Captura de pantalla del formulario de ingreso cuando se ingresa en duolingo. Es un ejemplo perfecto del uso de distintos tipos de cajas de texto y de la distinta programación de cada una de ellas.



### 1.2.2. Utilización de campos y textos

Si algo debería tener claro el alumno a estas alturas es que los formularios son un elemento de por sí, que se compone a su vez de otros elementos que dan forma y componen los distintos campos del formulario.

Un formulario programado sólo con sus etiquetas básicas, además de no servir para nada, tendría un aspecto visual totalmente opaco de cara al usuario final, que realmente tendría un elemento en la página web que no sería de utilidad, ni se visualizaría.

En todo caso, al igual que ocurre con la programación de otros elementos, y por supuesto en la metodología general de los programadores, es muy importante que el programador tenga claro los elementos que tiene un formula-

## UD1

rio que programa, que formará los campos que, finalmente, se encontrará el usuario final en la pantalla.

Para ello, se debe obtener una metodología de programación, que ya de por si es importante en la programación normal, pero que cobra vital importancia en la elaboración de formularios.

Por ejemplo, en los siguientes formularios, los campos se identifican más fácilmente en uno que en otro.

Ejemplo 1:

```
<html><head><title>Ejemplo de programación de un formulario.</title></head><body><form action="envio_imagenes.php" method="post" accept="image/gif,image/jpeg">Manda la imagen jpg o gif:<input type="file" name="imagen"/><br/><input type="submit" value="Mandar Imagen"/></form></body></html>
```

Ejemplo 2:

```
<html>

<head>
    <title> Ejemplo de programación de un formulario.</title>
</head>

<body>
    <form      action="envio_imagenes.php"      method="post"
accept="image/gif,image/jpeg">

        Manda la imagen jpg o gif:
        <input type="file" name="imagen" />
        <br/>
        <input type="submit" value="Mandar Imagen"/>
    </form>
</body>
</html>
```

El código de ambos ejemplos es idéntico, y el resultado obtenido al ver el resultado en un navegador es similar, pero a la hora de identificar los campos de un formulario en el primer caso es muy complicado identificarlos y en el segundo se hace con mucha facilidad.

Por lo tanto, algo que debe tener muy presente el programador cuando se enfrenta a un formulario es dividir en distintas líneas los distintos elementos que componen el formulario. No solo por la facilidad a la hora de obtener una visual del programa, sino también por la comodidad que supone a la hora de revisarlo cuando se enfrente con el código más adelante el mismo programador, o incluso un programador distinto que tenga que modificarlo.

Otro detalle importante que se observa en las anteriores líneas de código son las tabulaciones. El hecho de que el distintivo código se encuentre en cascada, con tabulaciones cuando están a menor nivel, no supone ningún tipo de modificación al código, pero sin embargo también sirven para saber, de una simple ojeada, que elementos están debajo de otros, o incluso contenidos entre dos etiquetas a un nivel superior.

La programación con etiquetas también trae un problema asociado a veces, y esto se ve acentuado en muchas ocasiones en la declaración de la etiqueta principal de un formulario, la etiqueta `<form>`.

Cómo ya hemos visto en esta Unidad Didáctica, se le pueden añadir muchos atributos, lo que puede ofuscar muchísimo el código, y que sea complicado de revisar.

En los casos que ocurra esto, el programador puede recurrir, si lo desea, a uno de los elementos de HTML que hace que evitar estos problemas sea relativamente sencillo, el que no discrimina entre mayúsculas y minúsculas.

Es decir, es lo mismo para un navegador este código:

```
<form action="formulario_de_alta.php" method="post"
enctype="application/x-www-form-urlencoded"
accept="image/jpeg" name="formulario_de_prueba" target="_blank" accept-charset="UTF-8" autocomplete="off"
novalidate>
```

Que este otro código:

```
<FORM ACTION="formulario_de_alta.php" METHOD="post"
ENCTYPE="application/x-www-form-urlencoded"
ACCEPT="image/jpeg" NAME="formulario_de_prueba" TARGET="_blank" ACCEPT-CHARSET="UTF-8" AUTOCOMPLETE="off"
NOVALIDATE>
```

## UD1

Por lo tanto, en el caso del código no quede claro, como en el primer ejemplo, el programador podría recurrir a este atributo del HTML. Incluso podría utilizarlo de forma habitual como costumbre cuando programe en general.



Hay que tener mucho cuidado cuando se programe indistintamente en mayúsculas y minúsculas ya que, aunque HTML sea un lenguaje que acepte una acepción u otra, no ocurre así con otros lenguajes. Esto es especialmente importante cuando se trabaje con PHP, ya que es un lenguaje que se mezcla con HTML, y puede llegar a dar problemas si el programador no tiene presente que elementos está usando en un lenguaje u otro.

---

En todo caso, el uso de todas estas herramientas, además de facilitarle el trabajo al programador y a todo aquel que tenga que revisar el código, y cumplir uno de los cuatro objetivos de la programación (corrección, claridad, eficiencia y portabilidad), sirve para identificar claramente los campos del formulario.

Y ojo, como campos del formulario se entienden todos aquellos elementos que lo componen, no sólo aquellos que pueden ser modificados por el usuario final, si no también todos aquellos que son relevantes, visualmente o no, para el mismo. Como por ejemplo, el texto plano.

Uno de los elementos más importantes del formulario, y a la vez uno de los grandes olvidados en ser explicados, es el texto plano.

El texto plano es, normalmente y casi siempre, la vía de información directa entre el programador y el usuario, dándole indicaciones concretas de lo que debe rellenar en cada campo de texto, seleccionar en los botones de opciones o elegir en un menú emergente.

Hoy en día algunos formularios optan por obviarlo, introduciendo elementos informativos dentro de los cuadros de texto, por ejemplo, pero la mayoría de formularios lo sigue usando como la principal forma de mostrar la información necesaria.

Por ejemplo, veamos una imagen que hemos visto unas páginas más atrás.

## UF1304: Elaboración de plantillas y formularios

The screenshot shows a Firefox browser window with the URL [https://accounts.google.com/SignUp?service=mail&hl=es&continue=http%3A%2F%2Fmail.google.com%2Fmail%2Fui%2Des-ha-envia-es-bl&utm\\_campaign=esIn](https://accounts.google.com/SignUp?service=mail&hl=es&continue=http%3A%2F%2Fmail.google.com%2Fmail%2Fui%2Des-ha-envia-es-bl&utm_campaign=esIn). The title bar says "Cuentas de Google". The main content is titled "Crea tu cuenta de Google". It features two sections: "Solo necesitas una cuenta" (which includes a note about accessing Google services with just a username and password) and "Personaliza Google a tu gusto" (which includes a note about customizing your profile and preferences). Below these are three small images of people: a woman, a man, and another woman. To the right is a large form with fields for "Nombre" (Name), "Nombre de usuario" (Username), "Contraseña" (Password), "Confirmar contraseña" (Confirm Password), "Fecha de nacimiento" (Birth Date), "Sexo" (Gender), "Teléfono móvil" (Mobile Phone), and "Tu dirección de correo electrónico actual" (Current Email Address). A "Iniciar sesión" (Sign In) button is at the bottom right.

Captura de pantalla del formulario de ingreso cuando se intenta hacer una nueva cuenta en gmail. Es un ejemplo perfecto del uso de distintos tipos de cajas de texto y de tamaños adecuados para cada información que se necesita. Además de otros elementos de formulario como son las listas de selección.

Aquí tenemos la pantalla del formulario de ingreso de google. En él podemos observar que, a pesar de contar con el ejemplo expuesto más arriba de informar en el cuadro de texto la información a llenar, todos los elementos del formulario están introducidos con un texto plano, maquetado de forma para que aparezca encima de cada elemento del formulario para que indique qué se tiene que llenar o seleccionar en él.

Aquí tenemos un ejemplo:

```
<html>
<head>
<title>Ejemplo de texto plano.</title>
</head>
<body>
<form action="formulario.php" method="post">
  Usuario:
  <input type="text" name="usuario" value="" /> <br/>
```

## UD1

**Password:**

```

<input type="password" name="contrasena" value="" />
<br/>

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

</form>

</body>

</html>
```

Como vemos, cada elemento del formulario está introducido mediante texto plano para identificar qué tiene que llenar el usuario en cada lugar.

El texto plano, por supuesto, se puede maquetar de distintas formas, o mostrar simplemente tal cual si ningún tipo de maquetación.

En esto, el lenguaje HTML le da múltiples opciones al programador, que podrá escoger la más adecuada.

Por ejemplo, tanto este código:

```

<html>

<head>

    <title>Ejemplo de la maquetación del texto plano.</title>

</head>

<body>

    <form action="formulario.php" method="post">

        Usuario:

        <input type="text" name="usuario" value="" />
        <br/>

        Password:

        <input type="password" name="contrasena" value="" />
        <br/>

    </form>
```

```

</body>

</html>

Como este código:

<html>

<head>

    <title>Ejemplo de la maquetación del texto plano.</title>

</head>

<body>

    <form action="formulario.php" method="post">

        <p>Usuario:<br/>
        <input type="text" name="usuario" value="" /></p>

        <p>Password:<br/>
        <input type="password" name="contrasena" value="" /></p>

    </form>

</body>

</html>

```

Tienen apariencias similares. Y, por supuesto, las opciones son casi infinitas. Por ejemplo se podrían usar las tablas HTML para introducir los elementos en su interior.

### 1.2.3. Etiquetas de los formularios

Los formularios, al igual que todos los elementos del código HTML, están declarados con etiquetas, que como ya hemos visto en esta unidad didáctica tienen distintos atributos que hacen que actúen de una forma o de otra.

Recapitulando, las etiquetas más importantes de un formulario son las siguientes:

## UD1

- Etiqueta FORM.

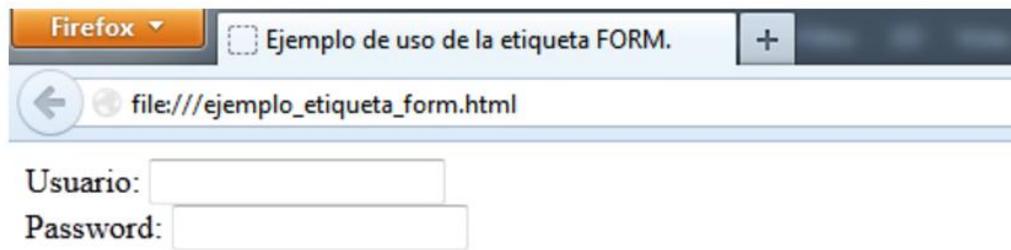
Es la etiqueta que delimita los formularios y que engloba a todos los elementos del mismo para poder relacionar que pertenecen al mismo formulario.

Por supuesto, como ya debería saber el alumno, es la etiqueta más importante del código HTML del mismo, y puede tener diversos atributos que le dan la funcionalidad que el programador quiera aplicarle a todos los elementos del formulario y a este en sí.

Aquí tenemos el código de un formulario, delimitado mediante la etiqueta form:

```
<html>
<head>
    <title>Ejemplo de uso de la etiqueta FORM.</title>
</head>
<body>
    <form action="formulario.php" method="post">
        Usuario:
        <input type="text" name="usuario" value="" />
        <br/>
        Password:
        <input type="password" name="contrasena" value="" />
        <br/>
    </form>
</body>
</html>
```

Captura de pantalla del resultado que debe dar el código al ejecutarlo en un navegador web.



– Etiqueta INPUT.

Es la etiqueta más importante para crear los elementos del formulario, ya que es la que engloba a la mayoría de elementos más relevantes del mismo, dependiendo de los atributos que se le asignen.

De hecho, con esta etiqueta, y como veremos más en profundidad en esta Unidad Didáctica, el programador puede crear:

- Botones de opciones.
- Campos ocultos.
- Campos de texto (oculto o no).
- Botones de Envío.
- Y muchas otras opciones.

Aquí tenemos el código de un formulario, con varias opciones con la etiqueta INPUT:

```
<html>
<head>
    <title>Ejemplo de etiqueta INPUT.</title>
</head>
<body>
    <form action=>>envio.php<> method=>>post<>>
        Nombre de la foto:

```

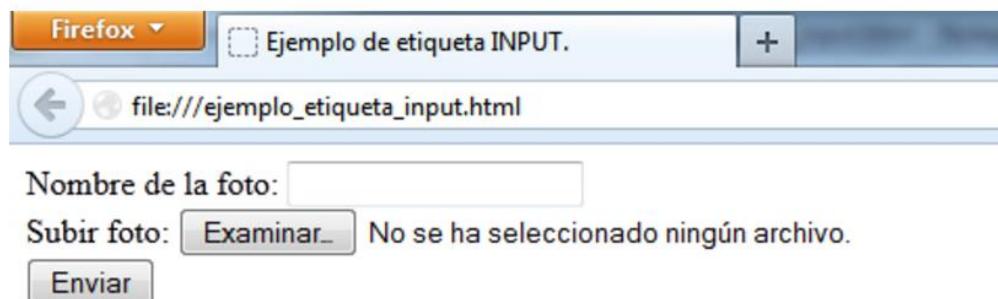
## UD1

```

<input type="text" name="nombre" value="" />
<br/>
Subir foto:
<input type="file" name="foto" />
<br/>
<input type="submit" value="Enviar" />
</form>
</body>
</html>

```

Captura de pantalla del resultado que debe dar el código al ejecutarlo en un navegador web.



- Etiqueta TEXTAREA.

Esta etiqueta se engloba dentro de las etiquetas específicas para los formularios, pero que son menos generales que las dos anteriores, las cuales, como hemos visto, son las realmente vitales para cualquier formulario.

En concreto, la etiqueta TEXTAREA se utiliza para definir o declarar un cuadro de texto más grande que una simple línea, como se hace con INPUT.

Los atributos y distintos ejemplos de esta etiqueta las veremos más adelante en esta Unidad Didáctica. Por ahora lo importante es quedarse con su función.

Veamos un ejemplo de la etiqueta TEXTAREA:

```
<html>

<head>
    <title>Ejemplo de uso de la etiqueta TEXTAREA.</title>
</head>

<body>

    <form action=>>formulario.php<> method=>>post<>>

        Usuario:
        <input type=>>text<> name=>>usuario<> value=>>> />
        <br/>

        Password:
        <input type=>>password<> name=>>contrasena<> value=>>> />
        <br/>

        Comentarios adicionales:
        <textarea name=>"comentarios"<> rows=>>2<> cols=>>55<>>
        </textarea>
        <br/>

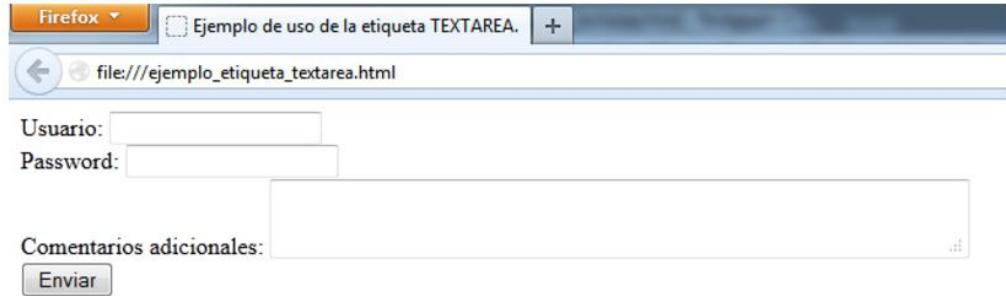
        <input type=>"submit"<> value=>"Enviar"<> />
    </form>

</body>

</html>
```

Captura de pantalla del resultado que debe dar el código al ejecutarlo en un navegador web.

## UD1



- Etiquetas SELECT y OPTION.

Dentro de las etiquetas menos relevantes, pero igualmente exclusivas y con mucha utilidad dentro de los formularios, nos encontramos con esta etiqueta SELECT.

Su función es la de crear una lista desplegable de elementos, que a su vez van a ser listados mediante el uso de la etiqueta OPTION.

Este elemento se usará cuando al programador le interese que el usuario tenga acotado el rango de la respuesta, en lugar de que tenga que introducir los datos manualmente.

Los distintos atributos de estas etiquetas se verán en profundidad más adelante en la Unidad Didáctica.

Veamos un ejemplo de ambas etiquetas:

```
<html>
<head>
    <title>Ejemplo de uso de las etiquetas SELECT y OPTION.</title>
</head>
<body>
    <form action=>color.php</action> method=>post</method>
        Seleccione el color que desea:
        <br/>
```

```

<select name="colores">

    <option value="blanco">Blanco</option>

    <option value="amarillo">Amarillo</option>

    <option value="verde">Verde</option>

    <option value="rojo">Rojo</option>

    <option value="azul">Azul</option>

</select>

<br/>

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

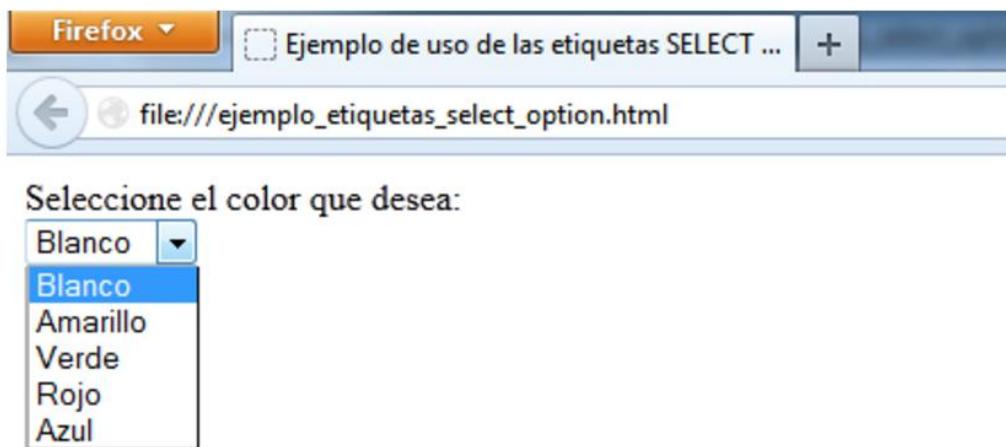
</form>

</body>

</html>

```

Captura de pantalla del resultado que debe dar el código al ejecutarlo en un navegador web.



- Etiqueta LABEL.

La etiqueta LABEL no es de por sí, al contrario que las anteriores, una etiqueta exclusiva de los formularios. Sin embargo, puede tomar una importancia capital en los mismos si se usa de forma correcta.

## UD1

Esta etiqueta representa la unión de un elemento a otro dentro del mismo formulario. Usándose especialmente para relacionar el texto plano con los elementos asociados situados dentro del formulario.

Este tipo de control se llama "control de etiquetado", y se puede relacionar al elemento que queremos asociar mediante la "id" del mismo.

Esto hace que pinchando en el texto concreto, el usuario tenga acceso rápido y visual al elemento que el programador ha enlazado.

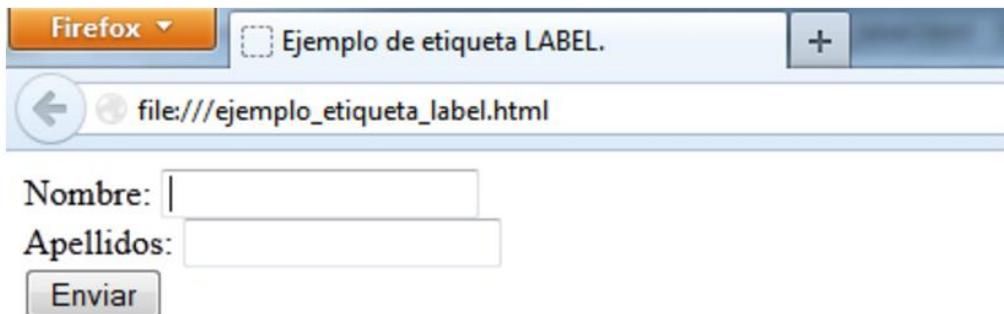
Veamos un ejemplo de la etiqueta LABEL:

```
<html>
<head>
    <title>Ejemplo de etiqueta LABEL.</title>
</head>
<body>
    <form action=>formulario.php</action> method=>post</method> name=>form_
envio_nombre</name>
    <label>
        Nombre:
        <input type="text" name="nombre" id="nombre" value="">
    </label>
    <br/>
    <label>
        Apellidos:
        <input type="text" name="apellidos" id="apellidos" va-
lue="" />
    </label>
    <br/>
    <input type="submit" value="Enviar" />

```

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

Captura de pantalla del resultado que debe dar el código al ejecutarlo en un navegador web.



En este ejemplo, si se hace click en el texto plano, el cursor irá inmediatamente al cuadro de texto asociado con la etiqueta LABEL.

- Etiquetas FIELDSET y LEGEND.

A veces agrupar los distintos campos dentro de un formulario no es suficiente para ordenarlos de forma deseada. Y la opción de agrupar los elementos en dos formularios no se puede contemplar. En esos casos es cuando entran estas etiquetas básicas de los formularios.

Cuando se quieren agrupar de forma lógica varios campos de un formulario se debe usar la etiqueta FIELDSET, que no es ni más ni menos que un elemento de bloque que agrupa de forma visual.

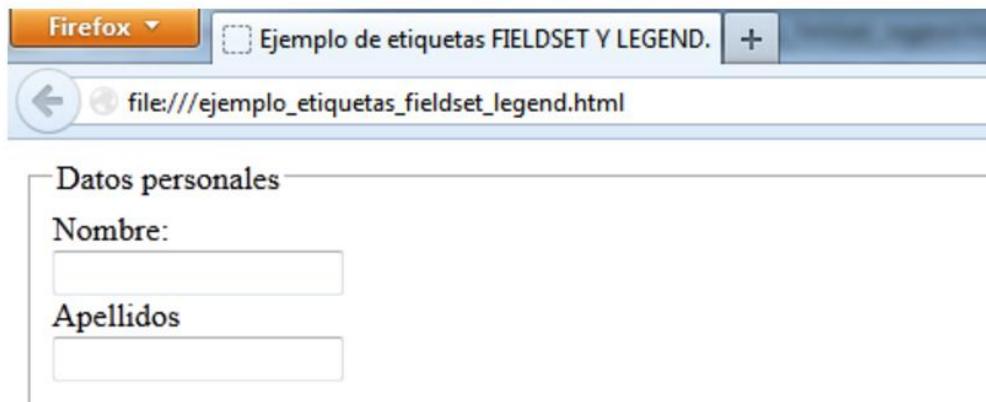
Cuando se quiere, además, definir un título o una leyenda para ese conjunto de campos agrupados mediante el uso de la etiqueta FIELDSET, se debe usar la etiqueta LEGEND.

## UD1

Veamos un ejemplo de ambas etiquetas:

```
<html>
  <head>
    <title>Ejemplo de etiquetas FIELDSET Y LEGEND.</title>
  </head>
  <body>
    <form action=>formulario.php</action> method=>post</method>
    <fieldset>
      <legend>
        Datos personales
      </legend>
      Nombre:
      <br/>
      <input type=>text</type> name=>nombre</name> value=></value>
      <br/>
      Apellidos
      <br/>
      <input type=>text</type> name=>apellidos</name> value=></value>
      <br/>
    </fieldset>
  </form>
</body>
</html>
```

Captura de pantalla del resultado que debe dar el código al ejecutarlo en un navegador web.



- Etiqueta BUTTON.

Aunque con la etiqueta INPUT efectivamente se pueden definir botones en el formulario. De hecho, como veremos más adelante, varios tipos de botones. Con la etiqueta BUTTON podemos definir botones para funciones no específicas por defecto dentro de la programación de un formulario.

Dentro de un botón definido con esta etiqueta se puede poner distinto tipo de contenido, como texto o imágenes, y esta es la principal diferencia con un botón creado mediante el elemento INPUT.

En todo caso, los distintos atributos del elemento BUTTON los veremos más detenidamente en esta Unidad Didáctica en su momento adecuado. Lo importante en estos momentos es que el alumno tenga claras las diferencias con la etiqueta INPUT.

A continuación tenemos un ejemplo de la etiqueta BUTTON:

```
<html>
<head>
    <title>Ejemplo de etiqueta BUTTON.</title>
</head>
<body>
    <form>
        <button name=>>enviar<> type=>>submit<> value=>>Mandar<>>></button>
    </form>
</body>
</html>
```

## UD1

```


</button>

<br />

<button name="resetear" type="reset" value="Borrar">
    Resetear
</button>

<br />

<button name="boton" type="button" value="deshabilitado"
disabled>
    Boton deshabilitado
</button>

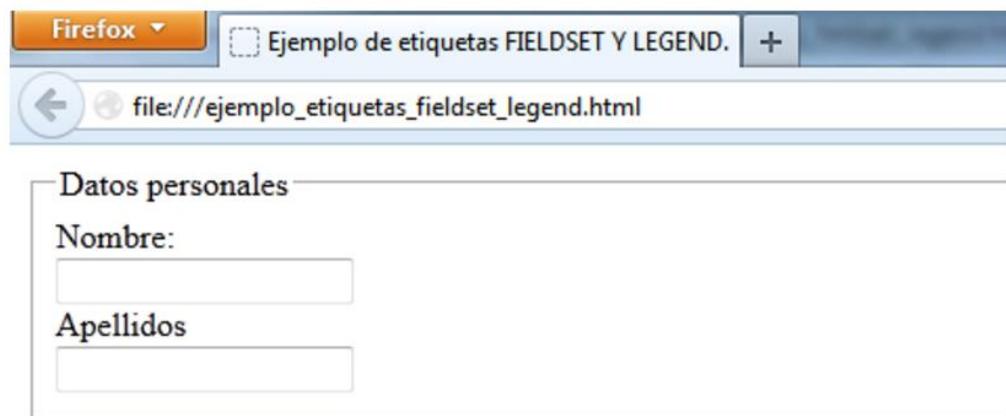
</form>

</body>

</html>

```

Captura de pantalla del resultado que debe dar el código al ejecutarlo en un navegador web.



Véase que en la captura del ejemplo no sale la imagen "enviar.png" si no el texto de alternativa, ya que no se encontró la imagen.

- Etiqueta DATALIST.

Con la etiqueta DATALIST se puede crear, ni más ni menos, un elemento INPUT con opciones parecidas a la etiqueta SELECT.

Esto es, cuando queramos, por ejemplo, poner varias opciones dentro de un campo de texto, donde el usuario puede introducir los datos que elija o, por el contrario, elegir entre unas opciones que introduzcamos.

Debido a esto, es la mejor opción cuando tenemos que darle al formulario una serie de opciones para elegir, pero estas son tan sumamente abiertas que, a veces, no se puedan recoger todos los casos de forma satisfactoria, solo los generales.

A continuación vemos un ejemplo de la etiqueta DATALIST:

```
<html>
<head>
    <title>Ejemplo de uso de la etiqueta DATALIST.</title>
</head>
<body>
    <form action=>color.php</action> method=>post</method>
        Seleccione el color que desea:
        <br/>
        <input list=>colores</list> name=>colores</name>
        <datalist id=>colores</id>
            <option value=>Blanco</value>
            <option value=>Amarillo</value>
            <option value=>Verde</value>
            <option value=>Rojo</value>
            <option value=>Azul</value>
        </datalist>
</form>
</body>
</html>
```

## UD1

```

<br/>

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

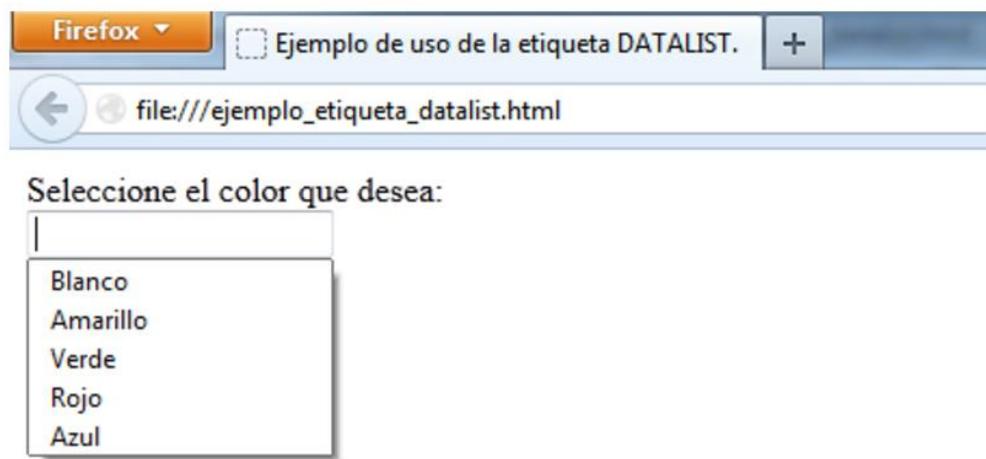
</form>

</body>

</html>

```

Captura de pantalla del resultado que debe dar el código al ejecutarlo en un navegador web.



- Etiqueta KEYGEN.

Esta etiqueta HTML es un poco especial, porque está específicamente diseñada para facilitar la generación de claves y el envío de las claves públicas como parte del formulario HTML.

Esto quiere decir que cuando se envía el formulario, la clave privada se almacena localmente, pero la clave pública es enviada al servidor. Por lo que se genera un par de claves gracias a este elemento del formulario.

Para entender completamente este elemento es necesario cierto conocimiento de la seguridad en el envío de paquetes y de los distintos algoritmos de seguridad que se utilizan. Pero al alumno ahora mismo solo le interesa conocer que existe este elemento, y saber en qué se usa.

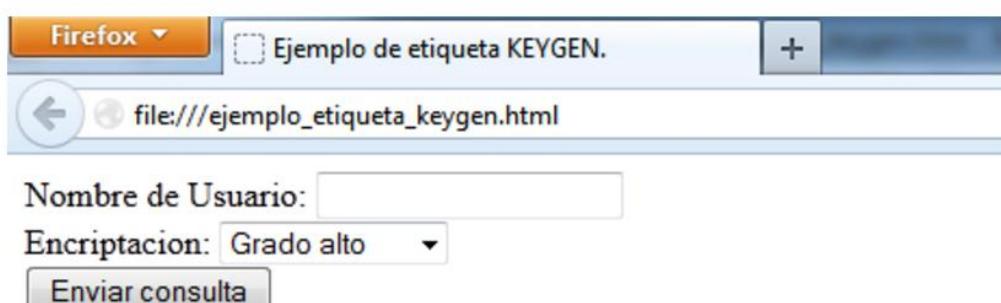
Veamos un ejemplo de la etiqueta KEYGEN:

```

<html>
  <head>
    <title>Ejemplo de etiqueta KEYGEN.</title>
  </head>
  <body>
    <form action=>keygen.php</action> method=>get</method>
      Nombre de Usuario:
      <input type=>text</type> name=>usuario</name>
      <br />
      Encriptacion:
      <keygen name=>seguridad</name>
      <br />
      <input type=>submit</type>
    </form>
  </body>
</html>

```

Captura de pantalla del resultado que debe dar el código al ejecutarlo en un navegador web.



## UD1

- Etiqueta OUTPUT.

Y por último, entre las etiquetas generales del formulario, nos encontramos con la etiqueta OUTPUT, que representa el resultado de un cálculo, ya sea efectuado por un script o en el propio formulario donde se encuentra.

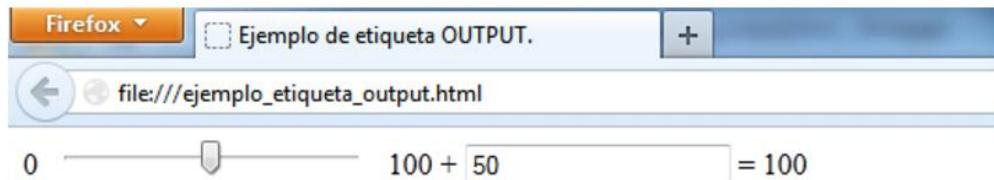
Con este elemento, se puede incluso especificar una relación entre los resultados del cálculo y los elementos usados en el mismo.

Además, en el caso de los formularios cobra más importancia, porque dependiendo de los datos introducidos por el usuario, se puede enviar una información u otra al servidor.

A continuación vemos un ejemplo de la etiqueta OUTPUT:

```
<html>
  <head>
    <title>Ejemplo de etiqueta OUTPUT.</title>
  </head>
  <body>
    <form oninput=>x.value=parseInt(a.value)+parseInt(b.value)>
      0
      <input type=>range id=>a value=>50>
      100 +
      <input type=>number id=>b value=>50>
      =
      <output name=>x for=>a b>>
    </output>
  </form>
</body>
</html>
```

Captura de pantalla del resultado que debe dar el código al ejecutarlo en un navegador web.



Dependiendo como se desplace la barra, se sumará más o menos, siendo el valor mínimo a sumar a 50 el cero, y el valor máximo el 100.

---

#### 1.2.4. Tamaños, columnas y filas de los formularios

Como ya hemos visto hasta el momento, los formularios pueden ser de distintos tamaños, desde el formulario más básico de búsqueda, como el del buscador de Google, hasta el más complejo de los formularios de la Agencia Tributaria.

Debido a ello, la presentación de un formulario es muy importante, no sólo por una cuestión estética, sino también por una mera cuestión de practicidad.

Un cuestionario demasiado escueto con poca información podría ser totalmente inservible para la función de la página web donde se programe.

Igualmente, un formulario demasiado extenso podría agotar al usuario. E incluso si los datos están demasiado apiñados puede resultar tan agreste que la mayoría de usuario simplemente ni lo completen por una cuestión de pérdida de paciencia.

Por lo tanto, el tamaño del formulario y cómo estén ordenados los datos cobra una importancia vital.

## UD1

Hasta ahora hemos visto formularios simples, donde las soluciones siempre han sido las más sencillas aparentemente. Pero para hacer un formulario, al igual que ocurre con el resto del código HTML, se pueden presentar los elementos de forma distinta:

- Usando los saltos de línea normales.
- Usando párrafos para separar los elementos.
- Usando una etiqueta para que el navegador muestre los datos tal y como están programados.
- Agrupando los elementos mediante etiquetas para ello.
- Metiendo los elementos dentro de tablas en código HTML.

En todo caso la elección debe ser siempre la más adecuada respecto a la relación de comodidad para el programador, coherencia con el resultado que necesita el usuario y cantidad de datos y elementos que tenga el formulario.

Por ejemplo, si entramos a la página web de la Agencia Tributaria y queremos hacer una encuesta de satisfacción de sus servicios nos encontramos con lo siguiente:

Captura de pantalla en el navegador Mozilla Firefox de la encuesta de satisfacción de la página web de la Agencia Tributaria en España.

Firefox - Encuesta de satisfacción

Encuesta de satisfacción

Con el objetivo de mejorar la calidad de nuestro servicio, le rogamos dedique unos momentos a completar esta pequeña encuesta, anónima y confidencial. Muchas gracias.

\* Campos obligatorios

• \* Motivo contacto:

- Renta
- IVA
- Sociedades
- Certificados
- Informativas
- Otros

• \* Calidad de trato:

- Mal
- Regular
- Bien
- Muy Bien

• \* Calidad de solución:

- Mal
- Regular
- Bien
- Muy Bien

• Comentario:

Tfnº. al que llamó:

Seleccione una opción ▾

• Si desea añadir datos relativos al agente que le atendió y la hora en la que contactó con nosotros, por favor indíquelo a continuación.

• Nombre agente: \_\_\_\_\_

• Fecha:

• 05 / Marzo / 2014

dia mes año

• Hora:

• hh : mm

hora minuto

Como se puede observar, en este formulario, además de los elementos que ya conocemos, la Agencia Tributaria ha optado por presentar un sistema de presentación algo antiguo, pero igual de efectivo, como es un sistema dividido en distintos puntos, relacionados entre sí, donde probablemente todo está programado en el mismo párrafo.

De cara al usuario esta programación es totalmente opaca. Sin entrar al código fuente, no podemos saber, normalmente, si el formulario está programado mediante saltos de línea normales, con párrafos para separar elementos o agrupados mediante diversas etiquetas HTML.

- Maquetación de Formularios mediante la etiqueta BR.

Este es el sistema de maquetación más simple y el que hemos usado hasta ahora en los ejemplos de forma general. Como el alumno ya debe saber a estas alturas, cuando se programa en código HTML, aunque se introduzcan espacios o saltos de línea en el código, el navegador lo interpreta como si todo estuviese escrito seguido salvo que se introduzcan las etiquetas adecuadas.

El introducir más espacios, o saltos de línea se hace, simplemente por limpieza de código y para facilitar las revisiones del mismo. Para introducir saltos de línea, y que estos se vean, se tiene que usar la etiqueta BR.

Vemos un ejemplo a continuación:

```
<html>
<head>
    <title>Ejemplo de maquetación de un formulario usando
    la etiqueta BR.</title>
</head>
<body>
    <form action=>>formulario.php<> method=>>post<>>
        Usuario:
        <input type=>>text<> name=>>usuario<> value=>>> />
        <br/>
        Password:

```

## UD1

```

<input type=>>password<> name=>>contrasena<> value=>>> />

<br/>

Repetir Password:

<input type=>>password<> name=>>confirmacion_contraseña<> va-
lue=>>> />

<br/>

Subir avatar de usuario:

<input type="file" name="avatar" />

<br/>

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

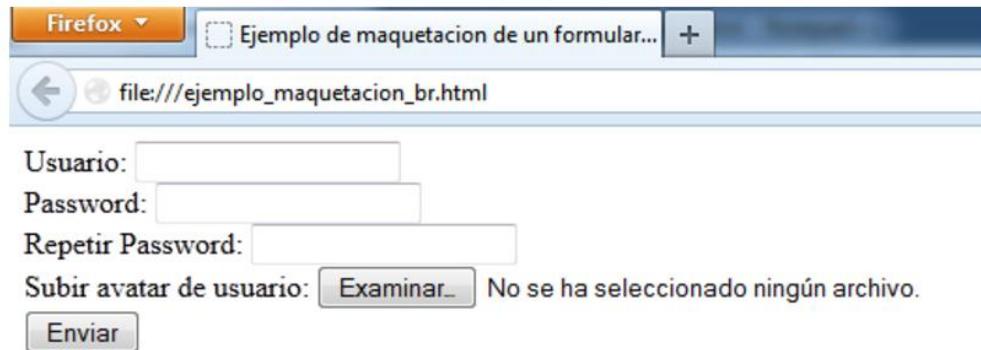
</form>

</body>

</html>

```

Captura de pantalla del resultado que debe dar el código al ejecutarlo en un navegador web.



- Maquetación de Formularios mediante la etiqueta P.

Aunque maquetar mediante la etiqueta BR es, sin duda, el sistema de maquetación más simple, como hemos visto en el punto anterior. No hay duda de que también se podía decir que es el menos elegante de todos.

Esto es totalmente opaco de cara al usuario, pero siempre es recomendable, de cara a seguir unos estándares de calidad, usar un sistema lo más correcto, y a la vez funcional, posible.

En concreto, los distintos elementos de un formulario se pueden distribuir usando la etiqueta `<P>`, que los distribuye en párrafos, y además evita efectos desagradables dependiendo del navegador y el tamaño de la resolución de pantalla que usen los distintos usuarios.

Vemos un ejemplo a continuación de la etiqueta `P`:

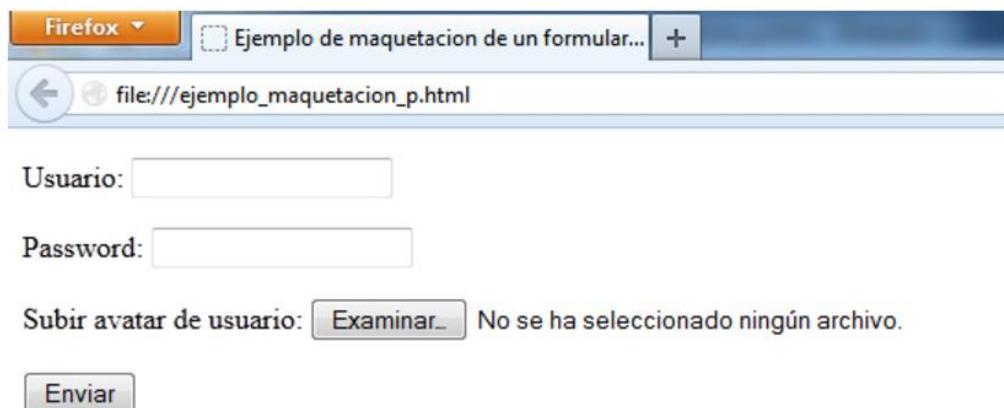
```
<html>
<head>
    <title>Ejemplo de maquetacion de un formulario usando
la etiqueta P.</title>
</head>
<body>
    <form action=>>formulario.php<> method=>>post<>>>
        <p>
            Usuario:
            <input type=>>text<> name=>>usuario<> value=>>> />
        </p>
        <p>
            Password:
            <input type=>>password<> name=>>contrasena<> value=>>> />
        </p>
        <p>
            Subir avatar de usuario:
            <input type=>>file<> name=>>avatar<> />
        </p>
    </form>
</body>
</html>
```

## UD1

```

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

Captura de pantalla del resultado que debe dar el código al ejecutarlo en un navegador web.



- Maquetación de Formularios mediante la etiqueta PRE.

Como ya sabe el alumno, en HTML no importa si introducimos más espacios o saltos de línea de la cuenta, ya que el navegador interpreta todo ello como si solo hubiese un espacio. Esto es muy útil cuando es necesario claridad de código, pero algo fastidioso a veces, cuando se pretende maquetar de forma visual.

Con la etiqueta PRE se elimina este problema. Cuando se usa, todo lo que está comprendido dentro de la etiqueta se ve tal cual está maquetado.

Esto es un arma de doble filo a veces, por lo que hay que tener cuidado cuando se tabula o se usan los saltos de línea para dar claridad al código, ya que con esta etiqueta se vería todo en el navegador.

Aquí tenemos un ejemplo de un formulario maquetado con la etiqueta PRE:

```
<html>
<head>
```

## UF1304: Elaboración de plantillas y formularios

```
<title>Ejemplo de maquetacion de un formulario usando  
la etiqueta PRE.</title>  
</head>  
<body>  
    <form action=>>formulario.php<> method=>>post<>>>  
        <pre>  
            Usuario:  
            <input type=>>text<> name=>>usuario<> value=>>> />  
            Password:  
            <input type=>>password<> name=>>contrasena<> value=>>> />  
            Repetir Password:  
            <input type=>>password<> name=>>confirmacion_contrasena<>  
            value=>>> />  
            Subir avatar de usuario:  
            <input type=>>file<> name=>>avatar<> />  
            <input type=>>submit<> value=>>"Enviar"<> />  
        </pre>  
    </form>  
</body>  
</html>
```

Captura de pantalla del resultado que debe dar el código al ejecutarlo en un navegador web.

## UD1

Firefox ▾ Ejemplo de maquetacion de un formulario... +

file:///ejemplo\_maquetacion\_pre.html

**Usuario:**

**Password:**

**Repetir Password:**

**Subir avatar de usuario:**

**Examinar...** No se ha seleccionado ningún archivo.

**Enviar**

- Maquetación de Formularios mediante la etiqueta SPAN.

La etiqueta SPAN no es, de por sí, una etiqueta con la que se puedan maquetar los elementos del formulario. Sin embargo, combinada con otras etiquetas para mostrar saltos de línea o párrafos, es una potente herramienta de maquetación, ya que permite ajustar lo que está contenido en sus etiquetas en función de la clase que se le asigne.

Esto en la práctica, puede hacer que parte de un formulario tenga un estilo u otro, lo que daría mucha más claridad y belleza visual al mismo.

Aquí tenemos un ejemplo el uso de la etiqueta SPAN para definir la clase de los elementos del formulario:

```
<html>
<head>
```

## UF1304: Elaboración de plantillas y formularios

```
<title>Ejemplo de maquetacion de un formulario usando  
la etiqueta SPAN.</title>  
  
</head>  
  
<body>  
  
<form action=>formulario.php</action> method=>post</method>  
  
<span class=>clase_de_prueba</span>  
  
    Usuario:  
  
    <input type=>text</type> name=>usuario</name> value=>''</value> />  
  
    <br />  
  
    Password:  
  
    <input type=>password</type> name=>contrasena</name> value=>''</value> />  
  
    <br />  
  
    Subir avatar de usuario:  
  
    <input type=>file</type> name=>avatar</name> />  
  
    <br />  
  
    <input type=>submit</type> value=>Enviar</value> />  
  
</span>  
  
</form>  
  
</body>  
  
</html>
```

Captura de pantalla del resultado que debe dar el código al ejecutarlo en un navegador web.

## UD1

- Maquetación de Formularios mediante la etiqueta DIV.

La etiqueta DIV sirve para, ni más ni menos, agrupar uno o más elementos a nivel de bloque, en este caso elementos de un formulario.

Esto es, que los elementos agrupados en etiquetas DIV distintas, no solo serán mostradas en párrafos aparte, sino que también pueden ser referenciadas en conjunto por otras herramientas, como por ejemplo con JavaScript.

A continuación vemos un ejemplo de un formulario maquetado usando la etiqueta DIV:

```

<html>
  <head>
    <title>Ejemplo de maquetacion de un formulario usando
    la etiqueta DIV.</title>
  </head>
  <body>
    <form action=>>formulario.php<> method=>>post<>>
      <div>
        Usuario:
        <input type=>>text<> name=>>usuario<> value=>>> />
      </div>
    </form>
  </body>
</html>

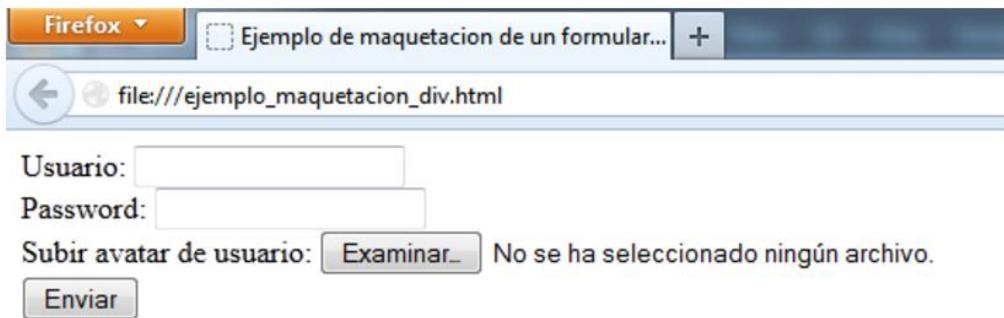
```

```

<div>
    Password:
    <input type=>password</> name=>contrasena</> value=></>
</div>
<div>
    Subir avatar de usuario:
    <input type="file" name="avatar" />
</div>
<div>
    <input type="submit" value="Enviar" />
</div>
</form>
</body>
</html>

```

Captura de pantalla del resultado que debe dar el código al ejecutarlo en un navegador web.



- Maquetación de Formularios mediante el uso de tablas.

Por último, hablamos de la que quizás es una de las herramientas más fiables para etiquetar un formulario. De hecho, lo es tanto, que es inconscientemente utilizada por muchos programadores cuando realizan un formulario con programas con plantillas de formularios.

## UD1

Por supuesto, hablamos de incluir los distintos elementos de un formulario en tablas, facilitando así su colocación en la pantalla cuando cargue en el navegador. Y, lo que es más importante, pudiendo realizar diseños que no se verían modificados por la anchura de la pantalla o la resolución del usuario final.

Aquí tenemos el ejemplo de un formulario maquetado usando las tablas en HTML:

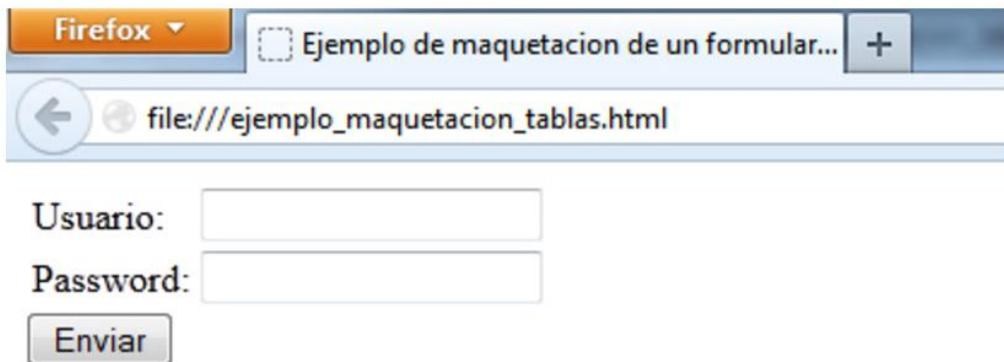
```

<html>
  <head>
    <title>Ejemplo de maquetacion de un formulario usando
    tablas en HTML.</title>
  </head>
  <body>
    <form action=>>formulario.php<> method=>>post<>>
      <table>
        <tr>
          <td>Usuario:</td>
          <td><input type=>>text<> name=>>usuario<> value=>>> />
        </td>
      </tr>
      <tr>
        <td>Password:</td>
        <td><input type=>>password<> name=>>contrasena<> value=>>>
        /></td>
      </tr>
      </table>
      <input type=>>submit<> value=>>Enviar<> />
    </form>
  </body>

```

```
</html>
```

Captura de pantalla del resultado que debe dar el código al ejecutarlo en un navegador web.



### 1.3. Controles de formulario

Los controles del formulario no son, ni más ni menos, que los elementos que el programador declara, usando etiquetas propias, comprendido dentro de la etiqueta del formulario, y que sirven, ni más ni menos, para dotar de las herramientas necesarias para que el programador pueda dotar al formulario de los campos deseados.

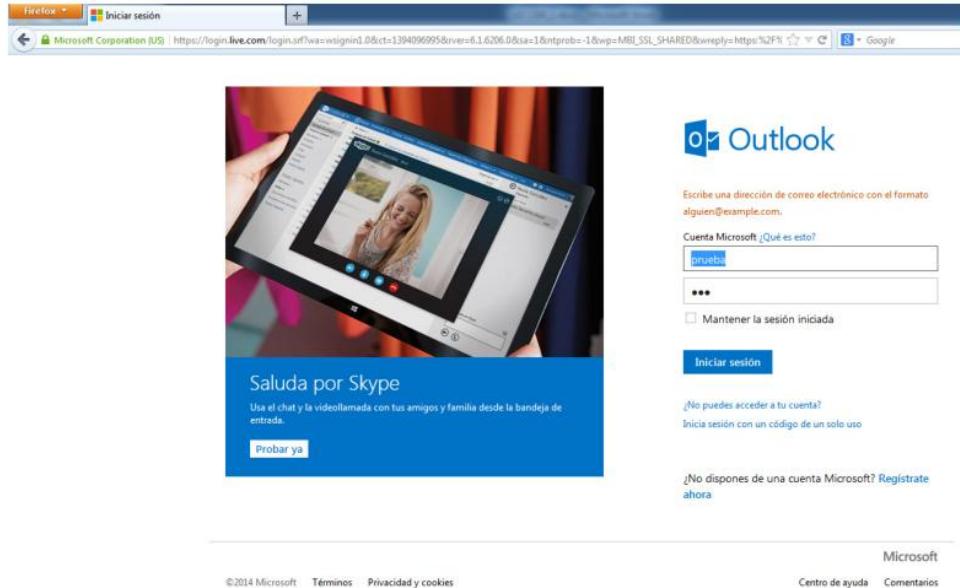
A los elementos del formulario se les denomina controles ya que sirven realmente para controlar el flujo de la información en su interior, y aunque son elementos que quedan para que sea el propio usuario, de forma interactiva, el que los rellene, en realidad el control que ejerce el programador sobre los datos que se pueden enviar es mucho mayor de lo que puede parecer a simple vista.

Por ejemplo, cuando un programador hace un formulario diseñado para enviar un correo, no tiene sentido que este se envíe sin una arroba entre sus caracteres, por lo que podría poner un sistema de control para evitar esto.

Veamos el siguiente ejemplo:

Captura de pantalla de la página principal del correo de Microsoft, la plataforma Outlook, cuando se intenta acceder al correo sin introducir una dirección de correo electrónico correcta sin arroba.

## UD1



Cuando se intenta acceder al correo de Outlook, que pertenece a Microsoft, da un error si no se introduce un correo en el formulario de acceso habilitado para ello. Por lo tanto el programador ha utilizado herramientas adecuadas para establecer un control del flujo de la información, que no es, ni más ni menos, que usar las opciones de ese control o que le brindan los lenguajes de programación de páginas web.



La arroba, el carácter por excelencia que va a asociada con el correo electrónico, fue elegida por el programador Ray Tomlinson debido a su significado en inglés, ya que en ese idioma la arroba se lee "at", que viene a significar "tal persona en tal lugar", por lo que es ideal para especificar el destinatario del mensaje.

---

Otro ejemplo claro de los controles de los formularios está en los caracteres máximos que pueden tener los campos según se los asigne el programador.

Por ejemplo, no tiene sentido que un campo de texto de edad tenga más de 3 caracteres, porque no hay nadie que tenga más de mil años.

## UF1304: Elaboración de plantillas y formularios

Aunque los casos más habituales donde se usan este tipo de control en los elementos del formulario, sin duda, es en la longitud de los campos de usuario y contraseña de registro en las páginas web, que debido normalmente a la disposición de las bases de datos no pueden aceptar nada más que un rango concreto de caracteres y unos valores específicos.

Por ejemplo, en la página web de Outlook, si se intenta realizar el registro de un correo, se puede ver un ejemplo de este caso concreto. Las contraseñas deben tener un número mínimo de caracteres.

Captura de pantalla de la página de registro del correo de Microsoft, la plataforma Outlook, en ella se observa claramente que las contraseñas deben tener como mínimo 8 caracteres.

Pero por supuesto, el hecho de poder establecer estos controles no quiere decir que los controles sean las restricciones del flujo de información. Los controles son, ni más ni menos, que los elementos del formulario, y lo son, como ya hemos dicho, porque permiten controlar el flujo de la información que va a enviar el formulario.

## UD1



Cuando se haga referencia a los elementos de un formulario, campos de un formulario o controles de un formulario, se estará refiriendo a lo mismo.

---

Por ejemplo, veamos el siguiente formulario:

```
<html>
  <head>
    <title>Ejemplo de formulario simple.</title>
  </head>
  <body>
    <form action=>>http://www.paginadeejemplo.com/formulario.php<> method=>>post<>>
      Nombre:
      <input type=>>text<> name=>>nombre<> value=>>> />
      <br/>
      <input type=>>submit<> value=>>Enviar<> />
    </form>
  </body>
</html>
```

En este formulario están establecidos claramente dos controles, además de una línea de texto plano.

El primero de los controles indica que el formulario, cuando se envíe, va a mandar la información que haya introducido el usuario final en ese cuadro de texto.

El programador ha indicado al usuario que debe llenar en ese cuadro su nombre, pero en la práctica no ha puesto ningún tipo de control con lo que se envía, por lo que en este caso, y sin tener en cuenta los controles que se puedan establecer más tarde en el lado del servidor, el usuario podría introducir lo que quisiese que esto sería enviado.

El segundo de los controles es un botón, de tipo submit para más señas, y al que además el programador le ha indicado mediante texto que se utiliza para enviar la información, que es, básicamente lo que se hace cuando el usuario pulsa ese botón.

En este ejemplo es muy sencillo aislar los elementos del formulario, así como el flujo de la información según la rellene el usuario, que va a depender, simplemente, de si rellena el campo de texto o no, y de cuando pulse el botón de enviar. Pero no siempre es tan sencillo localizar el flujo total de información, por lo que, como siempre, la claridad de código y saber muy bien cuál es el objetivo técnico y funcional es vital antes de empezar la programación en sí.

### 1.3.1. Descripción de los controles de los formularios

Los controles de los formularios, como todos los elementos en el lenguaje HTML, se definen usando etiquetas, que tienen distintos atributos que pueden ser o no declarados, y que además pueden contener a su vez otros elementos en su interior.

Cada uno de los controles tiene, como es lógico, una definición propia, pero la mayoría de ellos son creados mediante el uso de la etiqueta <INPUT>, que es un término en inglés que viene a significar entrada de datos.



Importante

En informática el sistema de entrada de información es muy importante, ya que indica una entrada o un cambio de información que se inserta en un sistema y que activa o modifica un proceso. A pesar de ser de por si un concepto abstracto, es en realidad uno de los más importantes en la informática.

---

De hecho, mediante la etiqueta input se pueden declarar:

## UD1

- Campos de texto.
- Campos de contraseñas.
- Listas de opciones mediante listas de cajas.
- Listas de opciones mediante botones de radio.
- Botones de enviar.
- Botones de borrar.
- Botones para subir archivos.
- Campos ocultos.
- Imágenes.
- Otro tipo de botones.

Sin embargo, estos no son los únicos controles que se pueden establecer dentro de un formulario. Mediante otros controles se pueden crear:

- Cuadros de texto de múltiples líneas con la etiqueta <TEXTAREA>.
- Uniones de un elemento a otro dentro del mismo formulario con la etiqueta <LABEL>.
- Agrupación de elementos para separarlos de forma visual con la etiqueta <FIELDSET>.
- Nombrar a las agrupaciones de elementos con la etiqueta <LEGEND>.
- Hacer una lista desplegable de opciones para ser seleccionadas por el usuario con la etiqueta <SELECT>.
- Hacer grupos dentro de las posibles selecciones de las listas desplegables con <OPTGROUP>.
- Definir opciones en las listas desplegables con la etiqueta <OPTION>.
- Declarar botones con otras funciones usando la etiqueta <BUTTON>.

- Poner varias opciones dentro de un campo de texto, donde el usuario puede introducir los datos que elija o, por el contrario, elegir entre unas opciones que introduzcamos con <DATALIST>.
- O mostrar el resultado de un cálculo usando la etiqueta <OUTPUT>.

En todo caso, cada uno de estos elementos debe ser usado cuando el programador requiera de su uso, y siempre dándole la funcionalidad requerida.

Veamos un ejemplo con todos los controles:

```
<html>
<head>
  <title>Ejemplo de formulario con todos los controles.</title>
</head>
<body>
  <form action="formulario.php" method="post">
    <fieldset>
      <legend>
        Controles del formulario
      </legend>
      <label>
        <input type="text" name="campo_de_texto" value="Este es un campo de texto" /><br/>
      <input type="password" name="campo_de_contrasena" value="Contrasena" /><br/>
      <input type="checkbox" name="opciones_cajas" value="Una opcion con botones de tipo caja" /><br/>
      <input type="radio" name="opciones_radio" value="Una opcion con botones de tipo radio" /><br/>
      <input type="submit" name="boton_envio" value="Boton de Enviar" /><br/>
    </fieldset>
  </form>
</body>
```

## UD1

```

<input type="reset" name="boton_borrar" value="Boton de Borrar" /><br/>

<input type="file" name="subir_archivo" value="Subir un archivo" /><br/>

<input type="hidden" name="campo_oculto" value="Este campo no se ve en pantalla" /><br/>

<input type="image" name="imagen" src="imagen.jpg" /><br/>

<textarea name="area_de_texto" rows="2" cols="55"></textarea><br/>

<select name="opciones">

    <optgroup label="Grupo de opciones 1">

        <option value="opcion11">Opcion 1.1</option>

        <option value="opcion12">Opcion 1.2</option>

    <optgroup label="Grupo de opciones 2">

        <option value="opcion21">Opcion 2.1</option>

        <option value="opcion22">Opcion 2.2</option>

    </select><br/>

    <button name="boton_desabilitado" type="button" value="boton_deshabilitado" disabled><br/>

    <input list="opciones" name="opciones">

    <datalist id="opciones">

        <option value="Opcion 1">
        <option value="Opcion 2">

    </datalist>

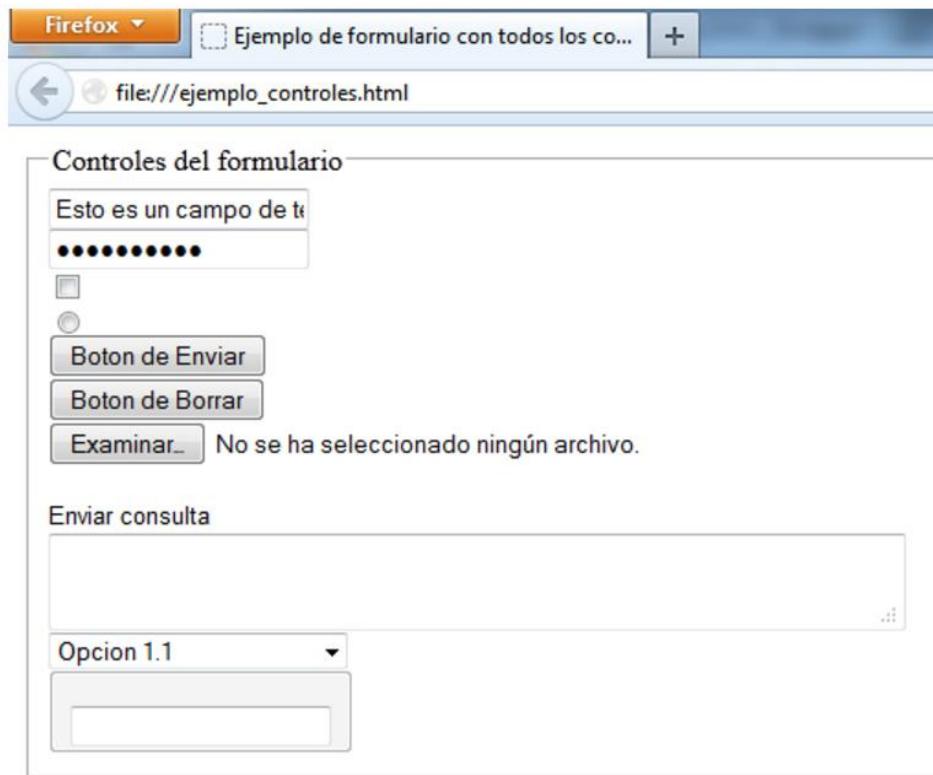
</label>
```

```

</fieldset>
</form>
</body>
</html>

```

Captura de pantalla del resultado que debe dar el código al ejecutarlo en un navegador web.



Obviamente este código no es, ni más ni menos, que una recopilación de los posibles controles que pueden aparecer en un formulario sin sentido técnico ni funcional.

Sin embargo, es muy importante que el alumno a estas alturas sepa ya reconocer el código y la representación gráfica de cada uno de los controles del formulario, no solo para que pueda programarlos con soltura en el futuro y esté familiarizado con ellos cuando sean visto más en profundidad en los próximos puntos, sino también para poder reconocer estos elementos cuando se encuentren ocultos u ofuscados en otras etiquetas HTML.

## UD1



El código ofuscado es aquel que se ha realizado tan intrincado, enrevesado, enredado y confuso que resulta prácticamente imposible su descifrado incluso el programador que lo creó. Crear este código no requiere técnicas criptográficas ni de complejos cifrados, sólo la pericia del programador, su imaginación y de su capacidad para programar de forma que, aunque parezca un follón sin sentido, quede de forma que funcione.

Un buen programador nunca intenta programar de forma ofuscada, salvo que pretenda evitar que su código sea reutilizado por otros programadores y esté haciendo un código que requiera poca revisión. Aún así estos casos deberían ser casi inexistentes.

---

A partir de ahora, en esta Unidad Didáctica vamos a ver los controles de los formularios en profundidad, con respectivos ejemplos para que el alumno pueda conocer todos y cada uno de los casos asociados a ellos.

### 1.3.2. Utilización de botones de acción

Como el alumno ya conoce, los formularios web se crearon partiendo como base de los formularios reales. Estos se crearon con la intención de poder recoger unos datos específicos de los que los llenaban, y se aplicó de la misma forma en los formularios web, con exactamente la misma función que los reales, recopilar información introducida por los usuarios.

Pero, si algo diferencia un formulario real de uno web es que, cuando terminas de llenar el primero, simplemente debes dárselo al receptor adecuado, y este se encargará de revisarlo, guardarlo y utilizarlo cuando sea conveniente, pero, ¿cómo se hace esa misma función con un formulario web? La respuesta está en un elemento nuevo, creado específicamente para estos, como son los botones.

Cuando un usuario termina de llenar un formulario web, tiene que tener alguna forma de enviarlo, ya que, físicamente, no puede entregarlo a su destino. Por lo tanto los botones cumplen una función esencial en los formularios web.

Sin embargo, no son los únicos botones que un programador puede crear. Están:

- Los botones creados con la etiqueta <INPUT> y el atributo SUBMIT.
- Los botones creados con la etiqueta <INPUT> y el atributo RESET.
- Los botones creados con la etiqueta <BUTTON>.

El primer atributo se utiliza para la función que ya se ha explicado, pero los otros dos cubren funcionalidades muy específicas y útiles también.

Cuando físicamente se está llenando un formulario y se comete un error, o simplemente se necesita un nuevo formulario del mismo tipo, la única opción es coger una hoja nueva de formulario y empezar a llenarla.

En los formularios web se podría hacer la misma opción, coger un nuevo formulario simplemente recargando la página, pero con el botón de reset, se puede reiniciar el formulario sin tener que hacer esto. Es más, concretando su funcionalidad se puede borrar sólo una parte del formulario, sin tener que recargar la página web entera.

Con la etiqueta <BUTTON> se cubren el resto de las opciones de botones que se pueden crear y servir de utilidad:

- Cuando se quiere crear un botón con un texto largo.
- Cuando se quiere crear un botón con una imagen.
- Cuando se quiere crear un botón para enviar parte de una información.
- Cuando se quiere crear un botón para borrar parte de un formulario, en lugar del formulario al completo.
- Cuando se quiere especificar ir a una URL al pulsar un botón.
- Cuando se quiere poner un botón fuera de un formulario pero ligarlo a un formulario concreto para enviarlo o borrarlo.
- Para marcar o desmarcar como válido un formulario.
- O incluso para mostrar los resultados tras la respuesta en un servidor de un formulario concreto.

## UD1

- Botón de envío usando el atributo INPUT con el tipo SUBMIT.

Este botón es el botón básico que todo formulario básico debe tener, ya que tiene la función más importante de todas, que es la de indicar cuando se debe enviar la información del formulario al servidor.

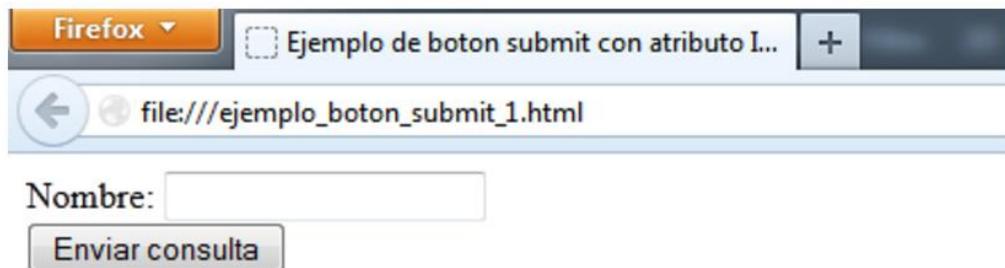
Cuando el usuario pulsa este botón envía los datos a donde se especifique en el atributo ACTION de la etiqueta FORM del formulario. Y se envían estos datos según se haya especificado en el atributo del método.

Para indicar que es este botón se debe indicar en el atributo TYPE de la etiqueta INPUT el valor "submit".

Veamos un ejemplo del botón:

```
<html>
  <head>
    <title>Ejemplo de botón submit con atributo INPUT.</title>
  </head>
  <body>
    <form action=>>envio_de_formulario.php<> method=>>post<>>
      Nombre:
      <input type=>>text<> name=>>nombre<> value=>>> />
      <br/>
      <input type=>>submit<> />
    </form>
  </body>
</html>
```

Captura de pantalla del resultado que debe dar el código al ejecutarlo en un navegador web.



Cuando se crea un botón con el elemento INPUT y el atributo SUBMIT, si no se le asigna ningún valor al atributo VALUE, coge el valor por defecto. En este caso ese valor es el de "Enviar consulta".

---

Como vemos en el ejemplo anterior, no es necesario asignar un valor al atributo VALUE, ni siquiera declararlo en la etiqueta INPUT, para programar un botón de envío. Sin embargo, es recomendable, no solo por funcionalidad, si no para orientar al usuario final, darle un valor adecuado que sea, además de visual, informativo.

Por ejemplo, a continuación hay un formulario con un valor del atributo VALUE adecuado para informar al usuario que va a ocurrir cuando pulse en el botón SUBMIT:

```
<html>
<head>
    <title>Ejemplo de boton submit con atributo INPUT.</title>
</head>
<body>
    <form action=>>alta_web.php</> method=>>post<>>>
        Usuario:

```

## UD1

```

<input type=>text</> name=>usuario</> value=></>
<br/>

Password:

<input type=>text</> name=>contrasena</> value=></>
<br/>

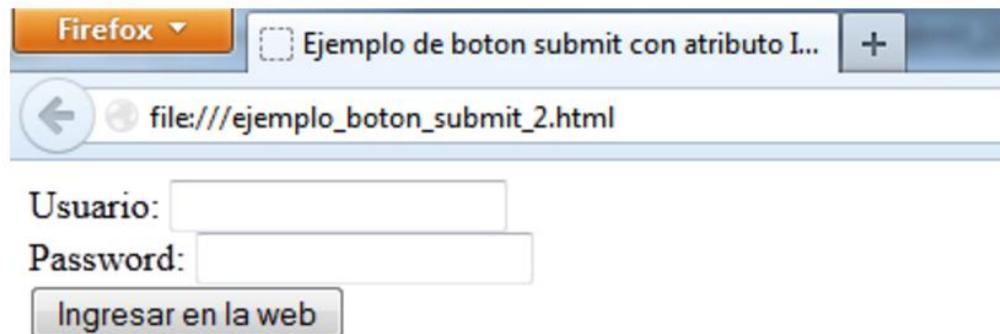
<input type=>submit</> value=>Ingresar en la web</>
</form>

</body>

</html>

```

Captura de pantalla del resultado que debe dar el código al ejecutarlo en un navegador web.



En este ejemplo, no solo se le informa al usuario que su formulario va a ser enviado, algo que en realidad debe ser totalmente opaco para él, ya que lo único que debe saber es que va a enviar la información que rellena y va a obtener el resultado que espera, sino que además le informa cuál es la función específica del botón, y que debe esperar cuando lo pulse, incluso antes de hacerlo.

Importante: No hay que olvidar que al pulsar el botón creado con la etiqueta INPUT y el atributo SUBMIT, se va a enviar todos los datos llenados en el formulario.

- Botón de envío usando el atributo INPUT con el tipo RESET.

Este botón, al contrario que el botón SUBMIT, debe ser muy cuidadosamente usado en los formularios, y debe ser específicamente indicado para que el usuario no lo pulse por error. De hecho, si se puede evitar ponerlo en diversas situaciones, mejor que mejor.

Cuando el usuario pulsa este botón borra todos los datos del formulario. Reinicializándolo de facto, haciendo así innecesario tener que recargar la página completa para ello.

Para indicar que es este botón se debe indicar en el atributo TYPE de la etiqueta INPUT el valor “reset”.

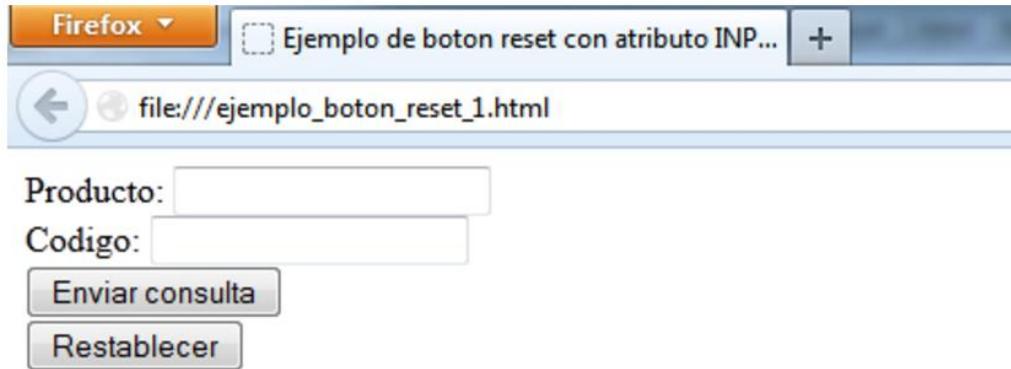
Veamos un ejemplo del botón:

```
<html>
<head>
    <title>Ejemplo de boton reset con atributo INPUT.</title>
</head>
<body>
    <form action=>>envio_de_producto.php<> method=>>post<>>>
        Producto:
        <input type=>>text<> name=>>producto<> value=>>>> />
        <br/>
        Código:
        <input type=>>text<> name=>>codigo<> value=>>>> />
        <br/>
        <input type=>>submit<> />
        <br/>
        <input type=>>reset<> />
    </form>
```

## UD1

```
</body>
</html>
```

Captura de pantalla del resultado que debe dar el código al ejecutarlo en un navegador web.



Cuando se crea un botón con el elemento INPUT y el atributo RESET, si no se le asigna ningún valor al atributo VALUE, coge el valor por defecto. En este caso ese valor es el de "Restablecer".

---

Como vemos en el ejemplo anterior, y al igual que ocurre cuando se programa un botón de tipo SUBMIT, no es necesario asignar un valor al atributo VALUE, ni siquiera declararlo en la etiqueta INPUT, para programar un botón de borrado. Sin embargo, es recomendable, no solo por funcionalidad, si no para orientar al usuario final, darle un valor adecuado que sea, además de visual, informativo. Especialmente en estos casos, que borrar toda la información de un formulario cuidadosamente rellenado puede ser realmente exasperante para el usuario final.

A continuación vemos el ejemplo de la página anterior, pero esta vez con el valor del atributo VALUE adecuado para informar al usuario que va a ocurrir cuando pulse en el botón RESET:

```

<html>
  <head>
    <title>Ejemplo de botón reset con atributo INPUT.</title>
  </head>
  <body>
    <form action=>>envio_de_producto.php<> method=>>post<>>>
      Producto:
      <input type=>>text<> name=>>producto<> value=>>> />
      <br/>
      Código:
      <input type=>>text<> name=>>codigo<> value=>>> />
      <br/>
      <input type=>>submit<> value=>"Enviar el producto al distribuidor" />
      <br/>
      <input type=>>reset<> value=>"Borrar todos los datos introducidos" />
    </form>
  </body>
</html>

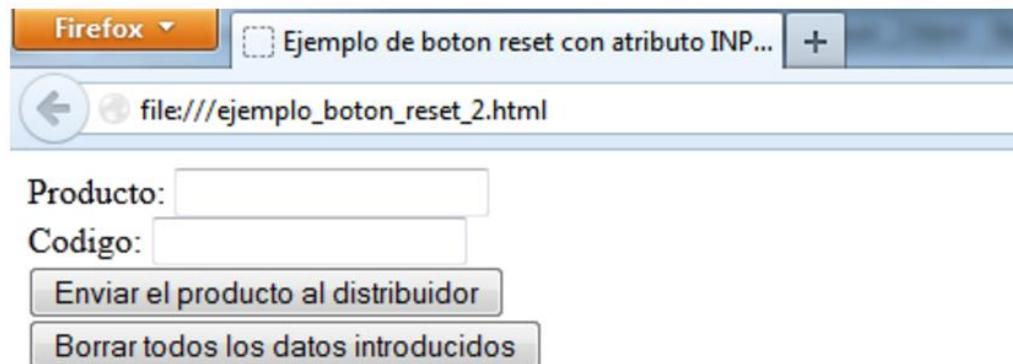
```

Captura de pantalla del resultado que debe dar el código al ejecutarlo en un navegador web.



No hay que olvidar que al pulsar el botón creado con la etiqueta INPUT y el atributo RESET, se van a borrar todos los datos rellenados en el formulario.

## UD1



En este ejemplo, no solo se le informa al usuario que su formulario va a ser restablecido, algo que igual no le acaba de quedar claro, sino que además le informa de que todos los datos van a ser borrados.

- Botón button.

Como ya hemos visto anteriormente, además de los dos botones básicos de los formularios creados con la etiqueta INPUT, se pueden crear botones para otras funciones específicas usando la etiqueta BUTTON.

Esta etiqueta no es solo utilizada para la creación de botones especiales que no pueden ser creados con las opciones por defecto, sino que también pueden ser utilizados por otros lenguajes como JavaScript o PHP para hacer referencia a líneas de código en concreto, o incluso para tratar con partes específicas del formulario en lugar de con todo el bloque.

A continuación vamos a ver los distintos atributos que se le pueden asignar a esta etiqueta, que a su vez servirá para que el alumno pueda ver que opciones ofrece.

Estos atributos son:

- Autofocus.
- Disabled.
- Form.
- Formaction.
- Formenctype.

- Formmethod.
- Formnovalidate.
- Formtarget.
- Name.
- Type.
- Value.

Algunos son muy parecidos a atributos ya aparecidos anteriormente en esta Unidad Didáctica, pero ligados a otros elementos. Igualmente los veremos en profundidad aplicados al controlador específico que estamos viendo.



Al contrario de lo que ocurre con el elemento INPUT, la etiqueta BUTTON debe ser cerrada con una etiqueta de cierre. Al ser botones muy similares no debe olvidarse esto, ya que si no se cierra el control afectará al resto del formulario englobándolo dentro de un botón.

---

En el caso del uso de la etiqueta BUTTON, se debe tener en cuenta que el aspecto final del botón puede variar dependiendo del navegador que utilice el usuario final.

Normalmente esto no tiene demasiada relevancia, ya que todos los navegadores soportan sin problema esta etiqueta. Pero cuando se hacen diseños específicos muy vistosos se debe tener cuidado con esto, ya que puede descuadrar todo el diseño.



Cuando se programa una página web es muy importante probarla en, al menos, los navegadores más utilizados por los usuarios.

---

## UD1

- Botón button con autofocus.

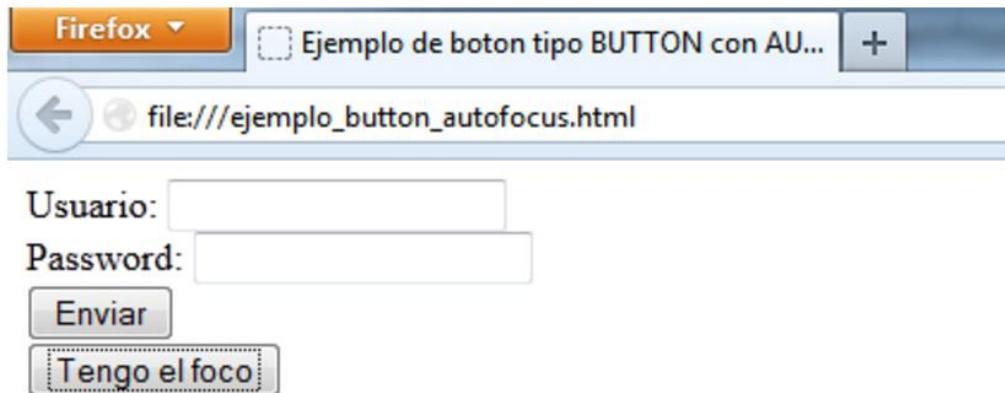
El primero de los atributos de la etiqueta BUTTON es el de AUTOFOCUS. Como su propio nombre indica, este atributo se utiliza cuando se quiere que el botón obtenga el foco cuando se cargue el formulario en el que está programado.

Esto es especialmente útil cuando se quiere que el botón esté resaltado por alguna razón en especial nada más cargar el formulario, o si se quiere proteger al usuario de equivocarse, haciendo que coja el foco un botón concreto.

Veamos un ejemplo de un formulario con un botón con este atributo:

```
<html>
<head>
    <title>Ejemplo de boton tipo BUTTON con AUTOFOCUS.</title>
</head>
<body>
    <form action="alta_web.php" method="post">
        Usuario:
        <input type="text" name="usuario" value="" />
        <br/>
        Password:
        <input type="text" name="contrasena" value="" />
        <br/>
        <input type="submit" value="Enviar"/>
        <br/>
        <button type="button" autofocus>Tengo el foco</button>
    </form>
</body>
</html>
```

Captura de pantalla del resultado que debe dar el código al ejecutarlo en un navegador web.



Véase como al cargar la página el foco lo tiene el botón con el autofocus.

- Botón button con disabled.

El atributo DISABLED es bastante particular.

Lo primero es que es un atributo booleano. Por lo tanto solo tiene dos posibles valores, estar activado o no. Para ello debe estar presente en la declaración del botón o no.

Lo segundo es que un botón con este atributo activo está deshabilitado, y no puede ser clicado hasta que se activa. Por lo tanto es un atributo con el que se debe jugar con otros lenguajes como con JavaScript o PHP.

Veamos un ejemplo de un formulario con un botón con este atributo:

```
<html>
<head>
  <title>Ejemplo de boton tipo BUTTON con DISABLED.</title>
</head>
<body>
  <form action=>>alta_web.php<> method=>>post<>>>>
    Usuario:
```

## UD1

```

<input type=>text</> name=>usuario</> value=></>
<br/>

Password:

<input type=>password</> name=>contrasena</> value=></>
<br/>

<input type=>submit</> value=>Enviar</>
<br/>

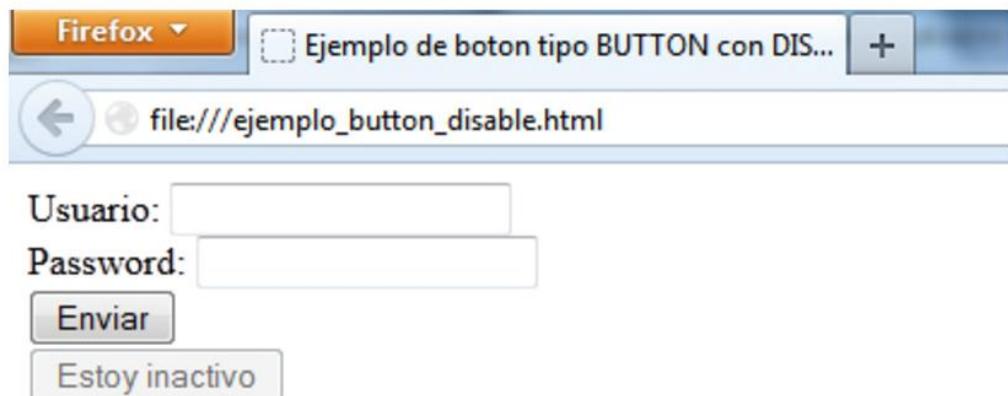
<button type=>button</> disabled>Estoy inactivo</button>
</form>

</body>

</html>

```

Captura de pantalla del resultado que debe dar el código al ejecutarlo en un navegador web.



Véase como al cargar la página el botón está desactivado.

- Botón button con form.

El atributo form tiene una utilidad muy importante, ya que permite que, a pesar de que un botón se declare en un lugar del código fuera del formulario, aún así pueda seguir referenciando a este.

Esto hace que podamos declarar un formulario en una parte de la presentación de una página web, mostrar mucha otra información, y donde quiera el programador poner un botón que refiera al formulario, por ejemplo para enviarlo.

Veamos un ejemplo de un formulario con un botón con este atributo:

```
<html>
<head>
    <title>Ejemplo de boton tipo BUTTON con FORM.</title>
</head>
<body>

    <form      action=>>alta_web.php</>      method=>>post<>
    id=>>formulario<>>

        Usuario:
        <input type=>>text<> name=>>usuario<> value=>>> />
        <br/>

        Password:
        <input type=>>text<> name=>>contrasena<> value=>>> />
        <br/>

        <input type=>>reset<> value=>>Borrar datos<>/>
    </form>
    <br/>
```

El formulario acaba en la linea anterior y a continuacion esta el boton de envio del formulario, que esta fuera del mismo.

```
<br/><br/>

    <button type=>>submit<> form=>>formulario<> value=>>Enviar
    formulario<>>Enviar formulario</button>

</body>
</html>
```

## UD1

Captura de pantalla del resultado que debe dar el código al ejecutarlo en un navegador web.



Véase como al cargar la página el botón está más abajo del formulario entre un elemento que no pertenece al mismo. Sin embargo, si se pulsa el botón, el formulario será enviado a la página de destino especificado en el atributo ACTION de la etiqueta FORM.

- Botón button con formaction.

El atributo formaction tiene una utilidad también muy interesante, ya que permite elegir dónde se van a enviar los datos del formulario, en lugar de ser enviados donde especifica el atributo ACTION de la etiqueta FORM.

Esto hace que podamos declarar un formulario que va a ser enviado a un lugar en concreto, pero podamos crear, por ejemplo, un botón para enviar el formulario a otro lugar específico.

Veamos un ejemplo de un formulario con un botón con este atributo:

```
<html>
  <head>
    <title>Ejemplo de boton tipo BUTTON con FORMACTION.</title>
  </head>
  <body>
    <form action=>alta_web.php</action method=>post</method>>
      Usuario:
    
```

```

<input type=>text</> name=>usuario</> value=></> />

<br/>

Password:

<input type=>text</> name=>contrasena</> value=></> />

<br/>

<input type=>submit</> value=>Enviar</>/>

<br/>

<button type="submit" formaction="administrador.php">Enviar los datos al administrador de la web</button>

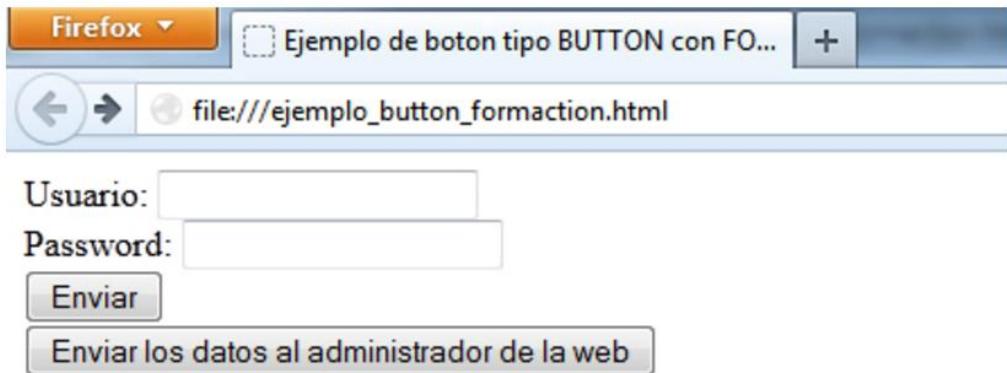
</form>

</body>

</html>

```

Captura de pantalla del resultado que debe dar el código al ejecutarlo en un navegador web.



Véase que si se le da al botón de enviar normal, la página de destino va a ser la especificada en el atributo ACTION de la etiqueta FORM, pero si se le da al botón con FORMACTION la página de destino va a ser la especificada en el botón.

## UD1

- Botón button con formenctype.

El atributo formenctype permite elegir que encriptación van a tener los datos del formulario cuando se envíen.

Esto hace que podamos declarar un formulario que va a ser enviado con una encriptación en concreto, pero podemos crear, por ejemplo, un botón para enviar el formulario con otra encriptación.

Como vimos anteriormente en la unidad didáctica hay tres tipos de encriptaciones:

- La "application/x-www-form-urlencoded", donde todos los caracteres son codificados antes de ser enviados (los espacios son enviados como "+" y los caracteres especiales son convertidos a ASCII hexadecimal).
- La "multipart/form-data", donde los caracteres no son codificados.
- Y la "text/plain", donde los espacios son enviados como "+", pero el resto de caracteres no son codificados.

Veamos un ejemplo de un formulario con un botón con este atributo:

```
<html>
<head>
    <title>Ejemplo de boton tipo BUTTON con FORMENCTYPE.</title>
</head>
<body>
    <form      action=>>alta_web.php<>      method=>>post<>
enctype=>>multipart/form-data<>>
        Usuario:
        <input type=>>text<> name=>>usuario<> value=>>> />
        <br/>
        Password:
        <input type=>>text<> name=>>contrasena<> value=>>> />
</body>
</html>
```

```

<br/>

<input type=>submit</> value=>Enviar</>

<br/>

<button type=>submit</> formenctype=>text/plain<>Enviar
como texto plano</button>

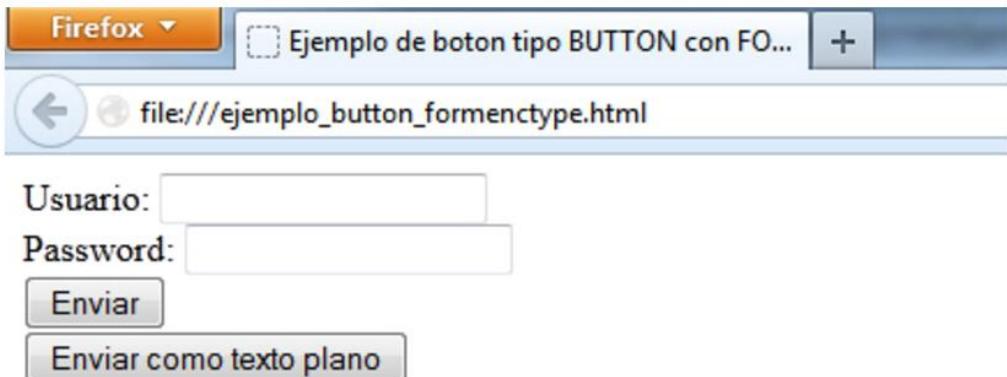
</form>

</body>

</html>

```

Captura de pantalla del resultado que debe dar el código al ejecutarlo en un navegador web.



Véase que si se le da al botón de enviar normal, se va a enviar con la codificación «multipart/form-data», y si se envía con el botón BUTTON se enviará como «text/plain».

- Botón button con formmethod.

El atributo formenctype permite elegir que método se va a utilizar para enviar los datos del formulario cuando se pulse el botón.

Esto hace que podamos declarar un formulario que va a ser enviado con un método en concreto, pero podemos crear, por ejemplo, un botón para enviar el formulario con el otro.

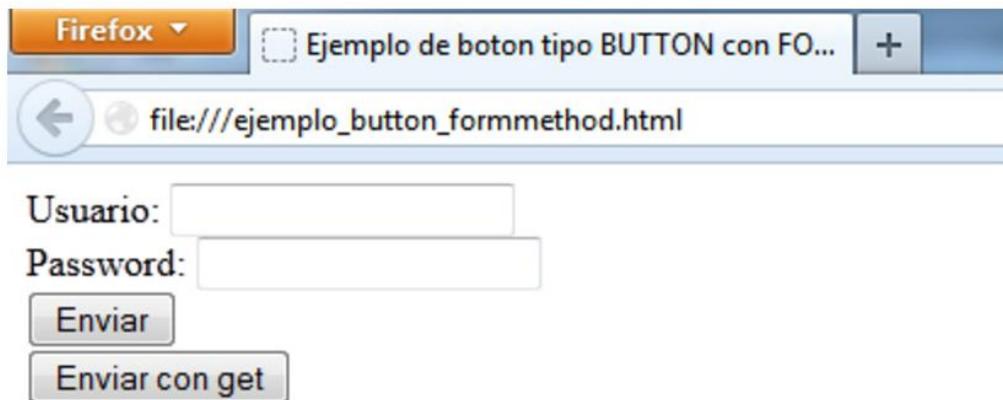
Cómo ya sabemos, hay dos métodos para enviar los formularios, el "get" y el "post", y cada uno tiene sus ventajas y sus inconvenientes.

## UD1

Veamos un ejemplo de un formulario con un botón con este atributo:

```
<html>
  <head>
    <title>Ejemplo de boton tipo BUTTON con FORMMETHOD.</title>
  </head>
  <body>
    <form action=>>alta_web.php<> method=>>post<>>>>
      Usuario:
      <input type=>>text<> name=>>usuario<> value=>>> />
      <br/>
      Password:
      <input type=>>text<> name=>>contrasena<> value=>>> />
      <br/>
      <input type=>>submit<> value=>>Enviar<>/>
      <br/>
      <button type=>>submit<> formmethod=>>get<>>Enviar con get</button>
    </form>
  </body>
</html>
```

Captura de pantalla del resultado que debe dar el código al ejecutarlo en un navegador web.



Véase que si se le da al botón de enviar normal, se va a enviar con el método "post", y si se envía con el botón BUTTON se enviará con el "get".

- Botón button con formnovalidate.

El atributo formnovalidate permite enviar el formulario sin que este esté validado. Esto es, actuar como el atributo NOVALIDATE, pero sólo si se pulsa el botón.

Esto hace que podamos declarar un formulario que normalmente cuando se pulsa el botón de enviar sea enviado normalmente, pero podemos crear, por ejemplo, un botón para enviar el formulario sin que este sea validado.

Veamos un ejemplo de un formulario con un botón con este atributo:

```
<html>
<head>
    <title>Ejemplo de botón tipo BUTTON con FORMNOVALIDA-
TE.</title>
</head>
<body>
    <form action=>>alta_web.php<> method=>>post<>>>
        Usuario:
        <input type=>>text<> name=>>usuario<> value=>>> />
    </form>
</body>
</html>
```

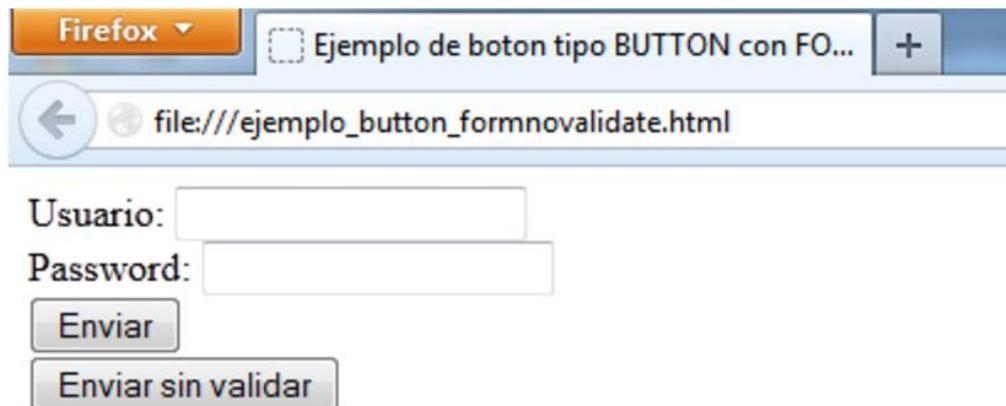
## UD1

```

<br/>
Password:
<input type=>text</> name=>contrasena</input> />
<br/>
<input type=>submit</> value=>Enviar</input>
<br/>
<button type=>submit</> formnovalidate>Enviar sin vali-
dar</button>
</form>
</body>
</html>

```

Captura de pantalla del resultado que debe dar el código al ejecutarlo en un navegador web.



Véase que si se le da al botón de enviar normal, se envía sin ningún problema, pero si se le da al botón de enviar con el FORMNOVALIDATE el formulario se envía sin validar.

- Botón button con formtarget.

El atributo formtarget permite especificar dónde se va a mostrar la respuesta del servidor una vez enviemos el formulario.

Esto hace que podamos declarar un formulario que normalmente presente los resultados en una ventana concreta, pueda presentarlo en otra dependiendo de la opción que tome el usuario.

Los valores que puede tomar el atributo formtarget son:

- \_blank: El resultado se muestra en una nueva ventana o pestaña.
- \_self: El resultado se muestra en la misma ventana que el formulario enviado.
- \_parent: El resultado se muestra en el marco "padre" del actual.
- \_top: El resultado se muestra a pantalla completa en la ventana.
- El nombre del marco: El resultado se muestra en el marco del nombre que se haya introducido (que se hará con texto plano).

Veamos un ejemplo de un formulario con un botón con este atributo:

```
<html>
<head>
    <title>Ejemplo de boton tipo BUTTON con FORMTARGET.</title>
</head>
<body>
    <form action=>>alta_web.php<> method=>>post<>>
        Usuario:
        <input type=>>text<> name=>>usuario<> value=>>> />
        <br/>
        Password:
        <input type=>>password<> name=>>contrasena<> value=>>> />
        <br/>
        <input type=>>submit<> value=>>Enviar<>/>
    </form>
</body>
</html>
```

## UD1

```

<br/>

<button type=><submit></button>>Mostrar resultado en otra pagina</button>

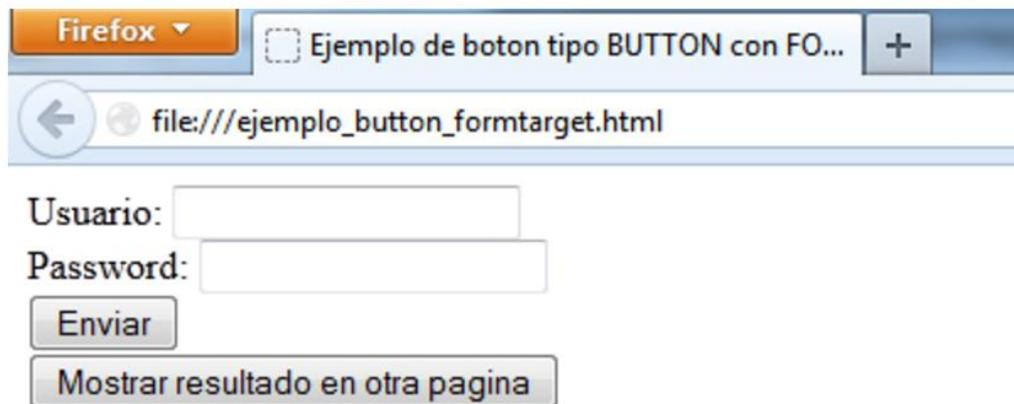
</form>

</body>

</html>

```

Captura de pantalla del resultado que debe dar el código al ejecutarlo en un navegador web.



Véase que si se le da al botón de enviar normal, se mostrará en la misma pantalla que es como actúa por defecto. Si se usa el botón con el FORMTARGET con "\_blank" se mostrará el resultado en una pantalla nueva.

- Botón button con name.

Al igual que ocurre normalmente con el resto de elementos en lenguaje HTML, el atributo NAME sirve para asignarles un nombre para poder ser referenciado por otros lenguajes como JavaScript.

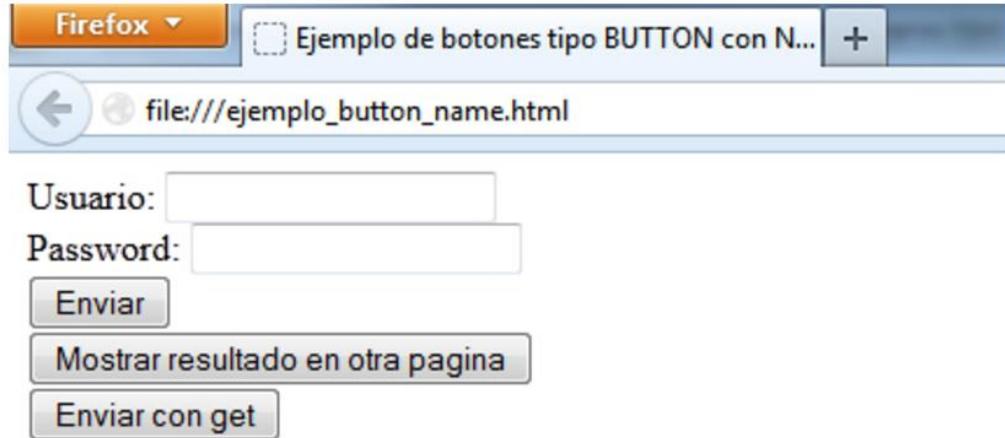
Realmente la funcionalidad de los atributos NAME e ID son parecidas, pero la ventaja de usar el atributo NAME es que se le puede dar a varios botones el mismo nombre, y cuando se programa en otro lenguaje y se hace referencia al elemento con ese nombre, todos los botones que tengan el mismo se verán afectados.

Veamos un ejemplo de un formulario con un botón con este atributo:

```
<html>
<head>
    <title>Ejemplo de botones tipo BUTTON con NAME.</title>
</head>
<body>
    <form action=>>alta_web.php<> method=>>post<>>
        Usuario:
        <input type=>>text<> name=>>usuario<> value=>>> />
        <br/>
        Password:
        <input type=>>text<> name=>>contrasena<> value=>>> />
        <br/>
        <input type=>>submit<> value=>>Enviar<>/>
        <br/>
        <button type=>>submit<> name=>>boton<> formtarget=>>_blank<>>Mostrar resultado en otra pagina</button>
        <br/>
        <button type=>>submit<> name=>>boton<> formmethod=>>get<>>Enviar con get</button>
    </form>
</body>
</html>
```

Captura de pantalla del resultado que debe dar el código al ejecutarlo en un navegador web.

## UD1



Véase que si se utilizase un código JavaScript que referenciase a los botones con el NAME "boton" afectaría a ambos botones por igual.

- Botón button con type.

Este es quizás el atributo más importante de todos los de la etiqueta BUTTON, ya que especifica el tipo del botón, y es esencial para saber su función.

Los posibles valores que puede tomar son:

- "button", cuando es un simple botón que actúa cuando se le hace click dependiendo de cómo esté programado.
- "submit", que indica que es un botón que envía la información del formulario.
- "reset", que indica que es un botón que borra la información del formulario.

Veamos un ejemplo de un formulario con un botón con este atributo:

```
<html>
<head>
    <title>Ejemplo de botones tipo BUTTON con distintos
    TYPE.</title>
</head>
<body>
```

```

<form action=>>alta_web.php<> method=>>post<>>

  Usuario:
  <input type=>>text<> name=>>usuario<> value=>>> />
  <br/>

  Password:
  <input type=>>text<> name=>>contrasena<> value=>>> />
  <br/>

  <button type=>>submit<>>Enviar el formulario</button>
  <br/>

  <button type=>>reset<>>Borrar datos</button>
  <br/>

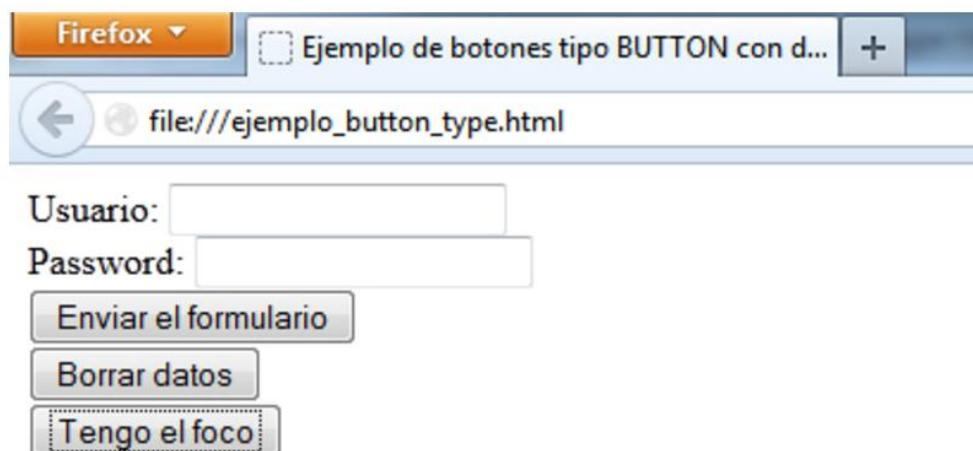
  <button type=>>button<> autofocus>Tengo el foco</button>
  </form>

</body>

</html>

```

Captura de pantalla del resultado que debe dar el código al ejecutarlo en un navegador web.



## UD1

Véase que hay un botón de cada tipo, con una función distinta cada uno.

- Botón button con value.

Y por último, está el atributo VALUE, que carece de utilidad con los navegadores actuales, pero que tiene gran importancia debido a que muchas veces no se puede saber con qué navegador va a entrar el usuario final.

Normalmente, cuando se declara un elemento tipo BUTTON, el valor que aparece cuando carga la web es el que se introduce entre las dos etiquetas.

Sin embargo, en navegadores más antiguos, e incluso en navegadores nuevos poco usados, el texto que va a mostrar es el del atributo VALUE. Por lo tanto es muy importante tenerlo en cuenta.

Veamos un ejemplo de un formulario con un botón con este atributo:

```
<html>
<head>
    <title>Ejemplo de boton tipo BUTTON con VALUE.</title>
</head>
<body>
    <form action=>alta_web.php</action> method=>post</method>
        Usuario:
        <input type=>text</type> name=>usuario</name> value=></value>
        <br/>
        Password:
        <input type=>text</type> name=>contrasena</name> value=></value>
        <br/>
        <input type=>submit</type> value=>Enviar</value>
        <br/>
```

```

<button type="button" value="Esto es el contenido de
value">Botón de prueba</button>

</form>

</body>

</html>

```

Captura de pantalla del resultado que debe dar el código al ejecutarlo en un navegador web.



Si se hubiese tratado de un navegador antiguo se habría mostrado el valor de VALUE en lugar del texto escrito entre las etiquetas.

Tras este tema el alumno debería tener claro cuántos tipos de botones se pueden encontrar en los formularios, y cuál es la utilidad de cada uno.

En definitiva, cualquier formulario siempre va a requerir el uso de al menos un botón, para indicar que se va a enviar la información del servidor. Y a partir de ahí, el número de posibilidades es tan alta como las necesidades reales del cliente y de la página web. Por lo tanto, son muchas.

En todo caso, a continuación vamos a ver un ejemplo con todos los tipos de botones que hemos visto en este punto de la Unidad Didáctica:

```

<html>

<head>

    <title>Ejemplo de todos los botones.</title>

</head>

```

## UD1

```

<body>

    <form      action=>alta_web.php>      method=>post<
id=>formulario<> enctype=>multipart/form-data<>>

        Usuario:

        <input type=>text<> name=>usuario<> value=><>> />

        <br/>

        Password:

        <input type=>text<> name=>contrasena<> value=><>> />

        <br/><input type=>submit<> value=>Enviar<>/>

        <br/><input type=>reset<> value=>Borrar todo<>/>

        <br/><button type=>button<> name=>boton<> autofocus>Tengo
el foco</button>

        <br/><button type=>button<> name=>boton<> disabled>Estoy
inactivo</button>

        <br/><button      type=>submit<>      name=>boton<>
form=>formulario<> value=>Enviar formulario<>>Enviar for-
mulario</button>

        <br/><button      type=>submit<>      name=>boton<>
formaction=>administrador.php<>>Enviar los datos al admi-
nistrador de la web</button>

        <br/><button      type=>submit<>      name=>boton<>
formenctype=>text/plain<>>Enviar como texto plano</button>

        <br/><button      type=>submit<>      name=>boton<>
formmethod=>get<>>Enviar con get</button>

        <br/><button      type=>submit<>      name=>boton<>
formnovalidate>Enviar sin validar</button>

        <br/><button type=>submit<> name=>boton<> formtarget=>_
blank<>>Mostrar resultado en otra pagina</button>

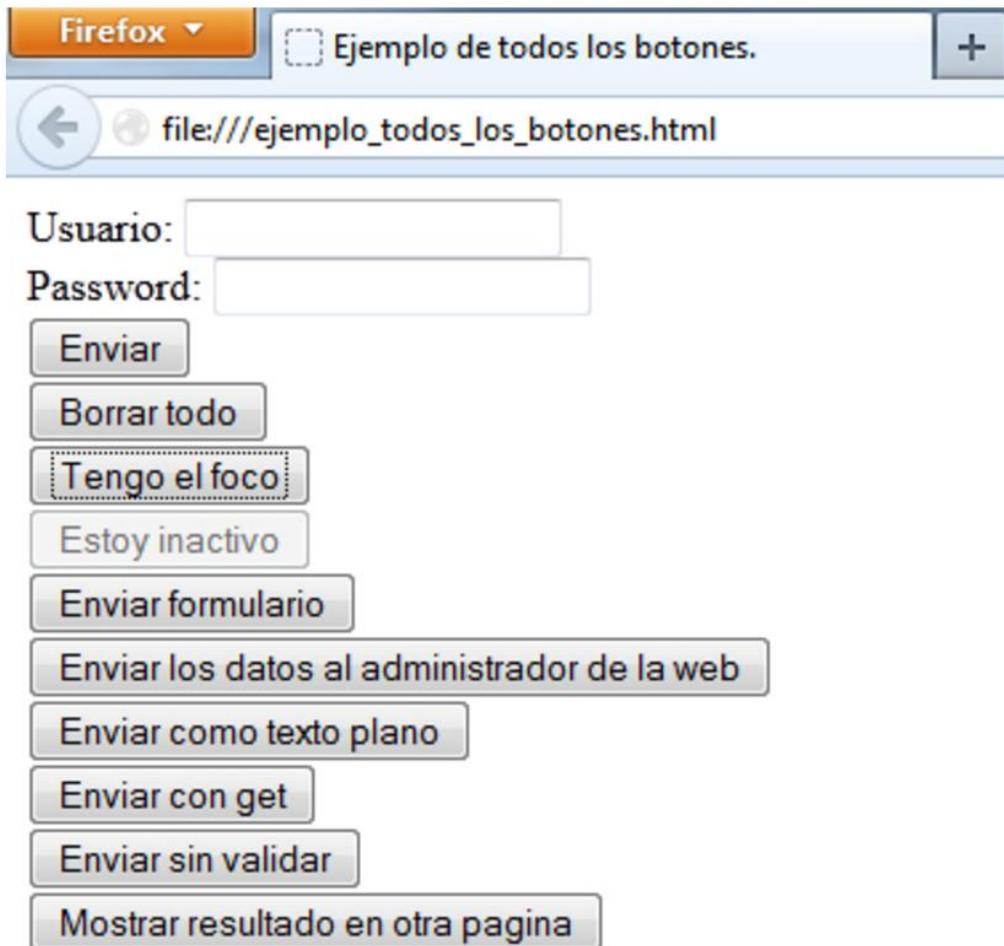
    </form>

</body>

</html>

```

Captura de pantalla del resultado que debe dar el código al ejecutarlo en un navegador web.



### 1.3.3. Utilización de lista desplegables

Las listas desplegables son, como su propio nombre indica, una serie de elementos listados que el usuario podrá seleccionar entre unas opciones que siempre son fijas.

Este elemento, al igual que ocurría con los botones, no existían, por razones lógicas, en los formularios reales. Pero su utilidad, ya que no pueden ser modificadas por el usuario y ofrecen siempre unos resultados más o menos fijos, que solo dependen de la opción que elija el usuario, y no de lo que introduzca

## UD1

manualmente, hace que sea una de las opciones más elegantes y útiles para representar diversos factores de los formularios, como son:

- Seleccionar entre unos datos fijos, como son los meses del año, países del mundo, provincias españolas, etc.
- Seleccionar datos en un rango determinado. Por ejemplo en lugar de pedir los ingresos se podría pedir que se especificase un rango salarial, o un rango de edad en lugar de pedir la edad.
- Acotar el número de resultados a un dato que se quiere saber fijo. Por ejemplo, si se pide la marca del coche podría acotarse la búsqueda a las marcas globales indicadas en una lista desplegable, evitando cientos de resultados distintos si fuesen los usuarios los que lo introdujesen manualmente.

Por ejemplo, en el formulario de registro de Facebook nos encontramos lo siguiente:

La captura de pantalla muestra la página de inicio de Facebook en español. En la parte superior, hay un banner azul con el logo de Facebook y el texto "Regístrate". Abajo de él, se presentan campos para ingresar nombre, apellido, correo electrónico y contraseña. A la derecha, hay un cuadro para "Fecha de nacimiento" que muestra un menú desplegable para seleccionar el mes. El menú desplegado muestra los meses del año: ene, feb, mar, abr, may, jun, jul, ago, sep, oct, nov, dic. Una opción "Hombre" está seleccionada. Al lado del menú, hay un aviso legal sobre las condiciones y políticas de privacidad.

Captura de pantalla del formulario de registro de la página web principal de Facebook vista en el navegador Mozilla Firefox.

El formulario en concreto, además de tener varios campos de texto para introducir el nombre, los apellidos, el email y la contraseña, tiene tres menús desplegables donde se puede seleccionar el día, mes y año de la fecha de nacimiento.



Facebook cuenta con más de 1.230 millones de usuarios, y que eso quiere decir que su formulario de registro ha sido utilizada, al menos ese número de veces.

Por supuesto, esto se puede hacer así porque se trata de datos fijos. Y es más cómodo tanto para el usuario, que no tendrá que introducir los datos a mano, como para el programador, que podrá trabajar siempre con los mismos datos de cara a almacenarlos en las bases de datos y trabajar con lenguajes como Oracle o MySQL.

Pero, por ejemplo, si se hubiese pedido un rango de edad, también podría haberse usado el sistema de las listas desplegables. Ya que, aunque la edad es un dato fijo, normalmente cuando se quiere saber la edad de una persona en un formulario se hace con la intención de saber la edad aproximada en la que se encuentra, no la exacta.

Veamos cómo es el formulario de búsqueda en la conocida web de empleo Infojobs:

La captura de pantalla muestra el formulario de búsqueda avanzada de Infojobs.net. En la parte superior, hay un cuadro de texto para 'Palabras clave' y un botón 'Búsqueda avanzada'. Debajo, se presentan los filtros principales:

- Categorías:** Una lista desplegable que incluye '(Todos)', Administración Pública, Administración de empresas, Atención a clientes, Comercio, retail e I+D, Comercial y ventas, Compras, logística y almacén, Diseño y arte gráficas, Educación y formación, Finanzas y banca, Informática y telecomunicaciones, Ingenieros y técnicos, Inmobiliaria y construcción, Legal, Marketing y comunicación, Profesiones, artes y oficios, Recursos humanos, Sanidad y salud, Turismo y restauración, Ventas al detalle y Políticas.
- Tus subcategorías:** Una lista desplegable que incluye 'Tus provincias'.
- Nivel de estudios:** Un selector que dice '(Indiferente)'.
- Experiencia mínima:** Un selector que dice '(Indiferente)'.
- Tipo de contrato:** Un selector que dice '(Todos)'.
- Jornada laboral:** Un selector que dice '(Seleccionar)'.
- Salario mínimo:** Un selector que dice '(Seleccionar)'.
- Antigüedad de las ofertas:** Un selector que dice 'Últimos 60 días'.

En la parte inferior derecha, hay un cuadro de texto que pregunta: '¿Quieres ver estadísticas (incluyendo salarios) de puestos de trabajo y empresas en España?' con un botón 'Sí'.

## UD1

Captura de pantalla del formulario de búsqueda de empleo de la página web de Infojobs vista en el navegador Mozilla Firefox.

Como se puede comprobar, en este formulario hay múltiples listas desplegables.

En este caso en concreto sirven para facilitar la tarea al buscador de empleo, ya que si, por ejemplo, tuviese que escribir a mano su categoría profesional, seguramente tendría más dificultades para encontrar ofertas, que haciéndolo directamente mediante un menú desplegable donde, tanto el que busca empleo como el que ofrece, han introducido los mismos valores que coincidirán en la búsqueda.

Por lo tanto, las listas desplegables son la mejor opción para este tipo de situaciones. Por lo tanto es vital conocerlas y dominar su programación a la hora de hacer mejores formularios.

En HTML la etiqueta para crear listas desplegables es <SELECT>, y como ocurre con la mayoría de etiquetas en este lenguaje debe abrirse y cerrarse, y todo lo que esté entre la etiqueta de apertura o de cierre se considera que está en la lista de atributos.

Sus atributos, que veremos más en profundidad a continuación, son los siguientes:

- Autofocus.
- Disabled.
- Form.
- Multiple.
- Name.
- Required.
- Size.

Cada uno de los valores que se introducen en una lista desplegable deben ser declarados como elementos también, al igual que ocurre con los elementos declarados dentro de un formulario.

En concreto, los elementos de una lista desplegable se declaran con la etiqueta <OPTION>, que también tiene etiqueta de apertura y de cierre.

Además, si se quieren agrupar varias opciones dentro de una lista desplegable se debe usar la opción OPTGROUP.

Este es un ejemplo de una selección simple:

```
<html>
  <head>
    <title>Ejemplo de lista desplegable simple.</title>
  </head>
  <body>
    <form>
      <select>
        <optgroup label=>Pitufos</optgroup>
        <option value=>Papa Pitufo</option>
        <option value=>Pitufina</option>
      </optgroup>
      <optgroup label=>Tortugas Ninja</optgroup>
      <option value=>Donatello</option>
      <option value=>Leonardo</option>
    </select>
  </form>
</body>
</html>
```

## UD1

- Lista desplegables SELECT con atributo autofocus.

El primer atributo que vamos a ver de la etiqueta SELECT es el de AUTOFOCUS. Dicho atributo se utiliza cuando el programador quiere que la lista desplegable obtenga el foco cuando se cargue el formulario en el que está programado.

Como suele ocurrir con este atributo en HTML, se trata de un atributo booleano, por lo que solo puede estar activo, cuando se encuentra declarado en la etiqueta, o inactivo.

Veamos un ejemplo de un formulario con una lista desplegable con este atributo:

```
<html>
<head>
    <title>Ejemplo de lista desplegable SELECT con AUTOFOCUS.</title>
</head>
<body>
    <form action="mandar_edad.php" method="post">
        Usuario:
        <input type="text" name="usuario" value="" />
        <br/>
        Edad:
        <select autofocus>
            <option value="menor">Menos de 18</option>
            <option value="adulto">De 18 a 65</option>
            <option value="jubilado">Mas de 65</option>
        </select>
        <br/>
        <input type="submit" value="Enviar"/>
    </form>
</body>
</html>
```

```

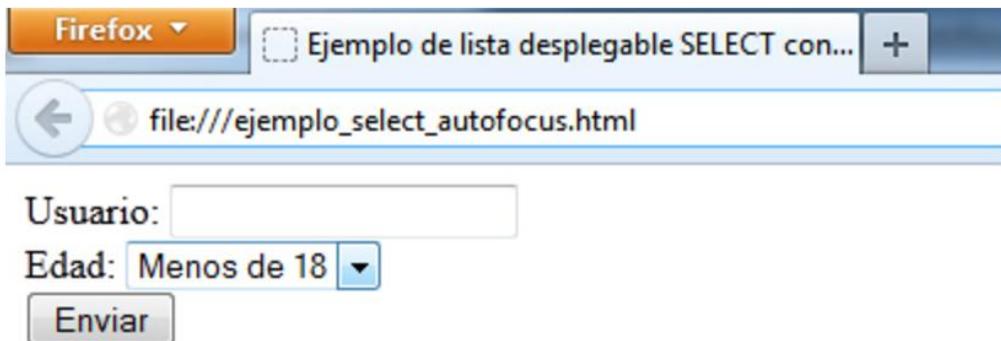
</form>

</body>

</html>

```

Captura de pantalla del resultado que debe dar el código al ejecutarlo en un navegador web.



- Lista desplegables SELECT con atributo disabled.

Cuando está presente en el elemento SELECT el atributo DISABLED, la lista desplegable se muestra deshabilitada, por lo que no se puede utilizar ni hacer click en ella. Al igual que el anterior, se trata de un atributo booleano, por lo que solo puede estar activo, cuando se encuentra declarado en la etiqueta, o inactivo.

Este atributo se puede utilizar junto con otros lenguajes como JavaScript para habilitar o deshabilitar la lista desplegable según se necesite.

Veamos un ejemplo de un formulario con una lista desplegable con este atributo:

```

<html>

<head>

    <title>Ejemplo de lista desplegable SELECT con DISABLED.</title>

</head>

<body>

```

## UD1

```

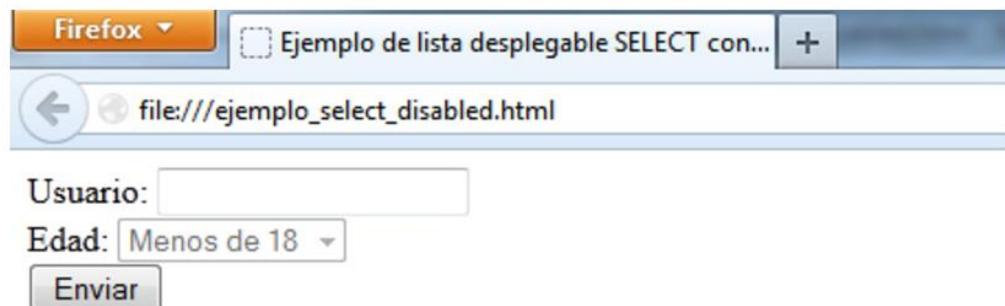
<form action=>>mandar_edad.php<> method=>>post<>>

    Usuario:
    <input type=>>text<> name=>>usuario<> value=>>> />
    <br/>

    Edad:
    <select disabled>
        <option value=>>menor<>>Menos de 18</option>
        <option value=>>adulto<>>De 18 a 65</option>
        <option value=>>jubilado<>>Mas de 65</option>
    </select>
    <br/>
    <input type=>>submit<> value=>>Enviar<>>/>
</form>
</body>
</html>

```

Captura de pantalla del resultado que debe dar el código al ejecutarlo en un navegador web.



- Lista desplegables SELECT con atributo form.

El atributo form en una lista desplegable tiene una utilidad muy importante, ya que permite que, a pesar de que una lista se declare en un lugar del código fuera del formulario, aún así pueda seguir referenciada a este.

Esto hace que podamos declarar un formulario en una parte distinta del código de una página web, programar otros elementos, y donde quiera el programador poner una lista desplegable que refiera al formulario.

Veamos un ejemplo de un formulario con una lista desplegable con este atributo:

```
<html>
<head>
    <title>Ejemplo de lista desplegable SELECT con FORM.</title>
</head>
<body>
    <form      action=>mandar_edad.php<>      method=>post<>
        id=>formulario<>

        Usuario:
        <input type=>text<> name=>usuario<> value=><> />
        <br/>
        <input type=>submit<> value=>Enviar<>/>
    </form>
    <br/>

    Este texto no forma parte del formulario.

    <br/><br/><br/>

    Edad del Usuario:
    <select form=>formulario<>
        <option value=>menor<>Menos de 18</option>
```

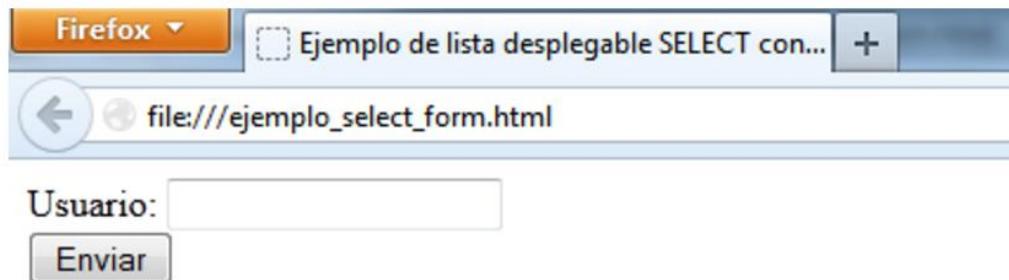
## UD1

```

<option value="adulto">De 18 a 65</option>
<option value="jubilado">Mas de 65</option>
</select>
</body>
</html>

```

Captura de pantalla del resultado que debe dar el código al ejecutarlo en un navegador web.



**Este texto no forma parte del formulario.**

**Edad del Usuario:** Menos de 18 ▾

- Lista desplegables SELECT con atributo multiple.

Normalmente en las listas desplegables sólo se puede seleccionar una de las opciones que ofrece. Esto es así debido a su funcionalidad, y que para mostrar múltiples opciones ya hay otras alternativas en los formularios. Sin embargo a veces el programador necesita que en una lista desplegable el usuario final pueda seleccionar varios de los objetos de la lista, ahí es donde entra este atributo.

El atributo MULTIPLE cuando se utiliza en el elemento SELECT permite, como su propio nombre indica, seleccionar múltiples opciones de la lista. Como es un atributo booleano, está activo cuando esté presente, e inactivo cuando no lo está.

Veamos un ejemplo de un formulario con una lista desplegable con este atributo:

```
<html>
<head>
    <title>Ejemplo de lista desplegable SELECT con MULTIPLE.</title>
</head>
<body>
    <form action=>>mandar_idiomas.php<> method=>>post<>>
        Usuario:
        <input type=>>text<> name=>>usuario<> value=>>> />
        <br/>
        Idioma que habla:
        <select multiple>
            <option value="castellano">Castellano</option>
            <option value="ingles">Ingles</option>
            <option value="aleman">Aleman</option>
        </select>
        <br/>
        <input type="submit" value="Enviar"/>
    </form>
</body>
</html>
```

Captura de pantalla del resultado que debe dar el código al ejecutarlo en un navegador web.

## UD1

Usuario:

Idioma que habla:

**Enviar**

- Lista desplegables SELECT con atributo name.

Al igual que ocurre normalmente con el resto de elementos en lenguaje HTML, el atributo NAME sirve para asignarles un nombre para poder ser referenciado por otros lenguajes como JavaScript.

Realmente la funcionalidad de los atributos NAME e ID son parecidas, pero la ventaja de usar el atributo NAME es que se le puede dar a varias listas desplegables el mismo nombre, y cuando se programa en otro lenguaje y se hace referencia al elemento con ese nombre, todas las listas que tengan el mismo se verán afectadas.

Veamos un ejemplo de un formulario con una lista desplegable con este atributo:

```
<html>
<head>
    <title>Ejemplo de lista desplegable SELECT con NAME.</title>
</head>
<body>
    <form action=>>mandar_edad.php<> method=>>post<>>>
        Usuario: 
        Idioma que habla: 
        <select name="idiomas">
            <option value="Castellano">Castellano</option>
            <option value="Ingles">Ingles</option>
            <option value="Aleman">Aleman</option>
        </select>
        <br/>
        <input type="submit" value="Enviar" />
    </form>
</body>
```

```

<input type=>text</> name=>usuario</> value=></> />

<br/>

Edad:

<select name=>edad</>

    <option value="menor">Menos de 18</option>

    <option value="adulto">De 18 a 65</option>

    <option value="jubilado">Mas de 65</option>

</select>

<br/>

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

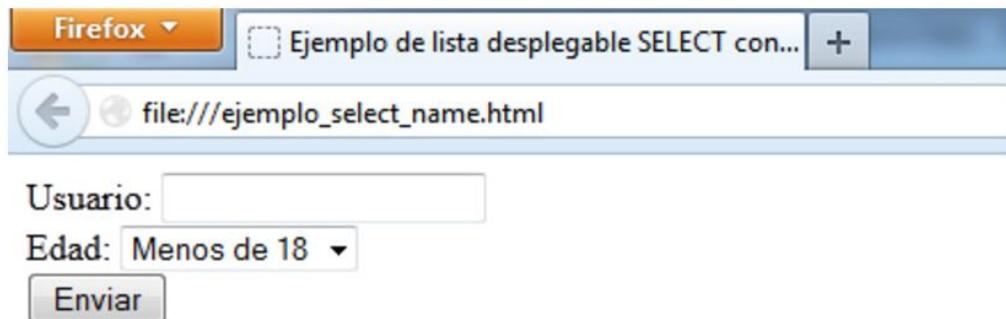
</form>

</body>

</html>

```

Captura de pantalla del resultado que debe dar el código al ejecutarlo en un navegador web.



- Lista desplegables SELECT con atributo required.

El atributo REQUIRED también es un atributo booleano, por lo tanto está activado si está presente o desactivado si no está.

## UD1

Cuando este atributo está presente quiere decir que el usuario final debe elegir un valor de la lista antes de enviar el formulario de forma obligatoria.

Veamos un ejemplo de un formulario con una lista desplegable con este atributo:

```

<html>
  <head>
    <title>Ejemplo de lista desplegable SELECT con REQUIRED.</title>
  </head>
  <body>
    <form action=>>mandar_edad.php<> method=>>post<>>
      Usuario:
      <input type=>>text<> name=>>usuario<> value=>>>> />
      <br/>
      Edad:
      <select required>
        <option value=>>menor<>>Menos de 18</option>
        <option value=>>adulto<>>De 18 a 65</option>
        <option value=>>jubilado<>>Mas de 65</option>
      </select>
      <br/>
      <input type=>>submit<> value=>>Enviar<>/>
    </form>
  </body>
</html>

```

Captura de pantalla del resultado que debe dar el código al ejecutarlo en un navegador web.

Firefox ▾

Ejemplo de lista desplegable SELECT con...

file:///ejemplo\_select\_required.html

Usuario:

Edad: Menos de 18

Enviar

- Lista desplegables SELECT con atributo size.

Al mostrar una lista desplegable normalmente por defecto se muestran cuatro de los elementos introducidos, aunque sea una lista de muchos más elementos.

Con el atributo SIZE, el programador puede indicar cuántos elementos de la lista desplegable van a mostrarse cuando el usuario interactúe con ella.

Ojo, si el tamaño que se le asigna a SIZE es mayor que el número de elementos, el navegador mostrará huecos en blanco para los elementos de más.

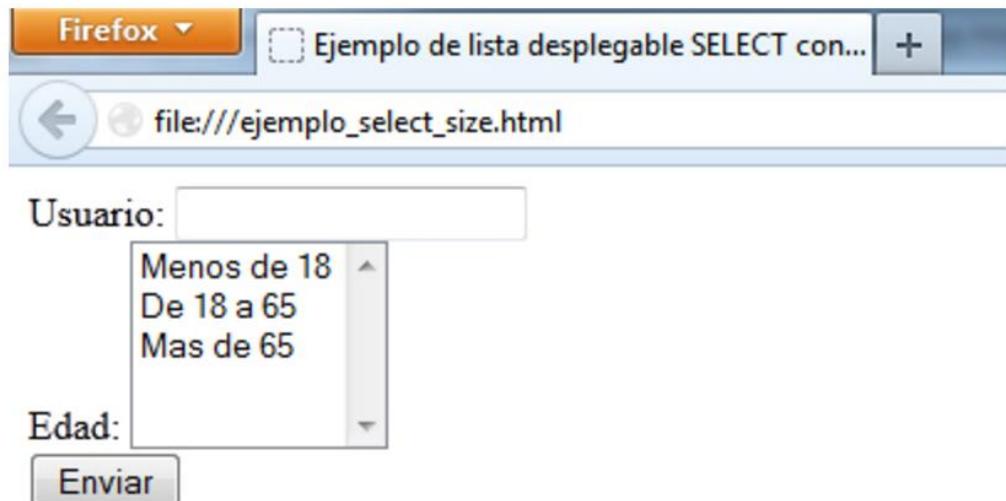
Veamos un ejemplo de un formulario con una lista desplegable con este atributo:

```
<html>
<head>
    <title>Ejemplo de lista desplegable SELECT con SIZE.</title>
</head>
<body>
    <form action=>>mandar_edad.php<> method=>>post<>>
        Usuario:
```

## UD1

```
<input type=>text</> name=>usuario</> value=></> />  
<br/>  
Edad:  
<select size=>5</>  
    <option value=>menor</>>Menos de 18</option>  
    <option value=>adulto</>>De 18 a 65</option>  
    <option value=>jubilado</>>Mas de 65</option>  
</select>  
<br/>  
<input type=>submit</> value=>Enviar</>/>  
</form>  
</body>  
</html>
```

Captura de pantalla del resultado que debe dar el código al ejecutarlo en un navegador web.



Tras este tema el alumno debería tener claro cuántos tipos de atributo puede encontrarse en las listas desplegables, y cuál es la utilidad de cada uno de ellos.

Cualquier formulario que requiera de un campo con una lista de opciones relativamente larga o acotada siempre va a requerir el uso de una lista desplegable, para facilitar el acceso a esos datos, ordenarlos, y facilitar la tarea tanto al programador como al usuario final. Incluso es posible crear una lista desplegable con múltiples opciones que pueden ser muy útiles en casos concretos.

En todo caso, a continuación vamos a ver un ejemplo con listas desplegables con todos los atributos que hemos visto en este punto de la Unidad Didáctica:

```
<html>
<head>
    <title>Ejemplo de lista desplegable SELECT con todos
    los atributos.</title>
</head>
<body>
    <form      action=>>mandar_datos.php<>      method=>>post<>
    id=>>formulario<>>
        Nombre: <input type=>>text<> name=>>usuario<> value=>>> />
        <br/>
        Edad:
        <select autofocus name=>>edad<> required>
            <option value=>>"menor"<>>Menos de 18</option>
            <option value=>>"adulto"<>>De 18 a 65</option>
            <option value=>>"jubilado"<>>Mas de 65</option>
        </select><br/>
        Idiomas:
        <select name=>>"idiomas"<> size=>>"3"<> multiple>
            <option value=>>"ingles"<>>Ingles</option>
            <option value=>>"frances"<>>Frances</option>

```

## UD1

```

<option value="aleman">Aleman</option>
</select><br />
<input type="submit" value="Enviar"/>
</form>      <br />
Opinion del formulario:
<select form="formulario" disabled>
<option value="buena">Buena</option>
<option value="regular">Regular</option>
<option value="mala">Mala</option>
</select>
</body>
</html>

```

Captura de pantalla del resultado que debe dar el código al ejecutarlo en un navegador web.

The screenshot shows a Firefox browser window with the title "Ejemplo de lista desplegable SELECT con...". The address bar displays "file:///ejemplo\_select.todos\_atributos.html". The form contains the following fields:

- Nombre:** A text input field.
- Edad:** A dropdown menu set to "Menos de 18".
- Idiomas:** A dropdown menu containing "Ingles", "Frances", and "Aleman".
- Enviar:** A submit button.
- Opinion del formulario:** A dropdown menu set to "Buena".

### 1.3.4. Utilización de casillas de verificación

Las casillas de verificación son elementos o controles del formulario que permiten al usuario final seleccionar opciones de forma individual. También son llamadas "checkbox".

Aunque a veces se muestran varias checkbox juntas, cada una de ellas es totalmente independiente del resto, ya que simplemente lo que hacen es mandar información si están marcadas o no cuando se envía el formulario al servidor.

Por lo tanto este elemento permite seleccionar al usuario varias opciones comunes pero no excluyentes entre sí.

Por ejemplo, en esta encuesta realizada por un usuario en Facebook nos encontramos que hace una pregunta y permite a los usuarios elegir varias opciones, sin que ninguna sea excluyente de las demás por elegirla.

**Which of my videos would you like to see more? :) I'm just curious!**

<input type="checkbox"/>	<b>Travel sketchbook</b>	...
<input type="checkbox"/>	<b>Watch me draw and paint</b>	...
<input type="checkbox"/>	<b>Giveaways</b>	...
<input type="checkbox"/>	<b>"How To" and tutorials</b>	...
<input type="checkbox"/>	<b>Studio tours</b>	
<input type="checkbox"/>	<b>Goofing around :)</b>	
<input type="checkbox"/>	<b>Other (Suggestions?)</b>	
<a href="#">+ Add an option...</a>		

Captura de pantalla de una encuesta con casillas de verificación en Facebook.

Sin embargo, cuando el programador quiere introducir una serie de opciones de forma similar a las casillas de verificación, pero de forma que sólo una de las opciones sea elegible, tiene que usar una herramienta similar a las casillas de verificación, como son los botones de opción.

Los botones de opción, también llamados botones de radio o "radiobutton" son muy parecidos a las casillas de verificación, pero la gran diferencia entre ellos es que todas las opciones presentadas por un botón de radio son ex-

## UD1

cluyentes entre sí. Es decir, sólo se puede elegir una opción entre todas las presentes.

Por ejemplo, en la siguiente encuesta realizada en el periódico El País, los usuarios sólo tienen la posibilidad de elegir una de las opciones.

The screenshot shows a poll interface from the website elpais.com. At the top, there's a header bar with the title "EL PAÍS - Elige el mejor músico español" and a link to "elpais.com/encuestas/html/2014/02/13/1392319636.html". Below the header, there are two tabs: "VOTACIÓN" (which is selected) and "RESULTADOS" (with "8760 respuestas"). The main area is titled "¿Cuál es para ti el mejor músico español?". A vertical list of 25 options follows:

- Radio Futura
- Joan Manuel Serrat
- Camarón
- Gabinete Caligari
- Vainica Doble
- Los Brincos
- Nacha Pop
- Alaska y Dinarama
- Andrés Calamaro
- Loquillo
- Burning
- Kiko Veneno
- Golpes Bajos
- Cánovas, Rodrigo, Adolfo y Guzmán
- Veneno
- Alaska y Los Pegamoides
- Los Planetas
- Triana
- Pata Negra
- El Último de la Fila
- Enrique Morente/Lagartija Nick
- Parálisis Permanente

Captura de pantalla de una encuesta en el periódico El País, sobre cuál es el mejor músico español para los lectores de ese periódico, realizada en el navegador Mozilla Firefox.

Por lo tanto, cuando se necesiten proporcionar varias opciones al usuario final, pudiendo este elegir cualquiera de ellas, la mejor opción será usar las casillas de verificación.

Sin embargo, si el usuario puede elegir sólo una opción entre la lista de elementos, la mejor opción sería realizarla con botones de opción.



Aunque en la práctica cualquier lista de opciones puede ser programada con casillas de verificación, ya que simplemente se pretende que el usuario elija una opción para enviar al servidor a través del formulario, si en la lista de opciones sólo se debe elegir una opción siempre se debe programar con botones de radio.

Igualmente, cuando se programen listas con los botones de radio se ha de ser muy cuidadoso, ya que si hay, al menos, dos de las opciones que podrían ser seleccionadas al mismo tiempo por el usuario, esto no será posible si están programadas como botones de opción.

---

En todo caso, la diferencia entre ambas opciones es sobre todo funcional, aunque no hay que obviar las diferencias técnicas, que también las tienen.

Veamos un ejemplo concreto del uso de ambas opciones en un formulario:

```
<html>
<head>
    <title>Ejemplo de casillas de verificacion y botones de opcion.</title>
</head>
<body>
    <form      action=>>mandar_datos.php</>      method=>>post</>
    id=>>formulario</>>
        Nombre:
        <input type=>>text</> name=>>usuario</> value=>>> />

```

## UD1

```

<br/><br/>

Rango Salarial:

<br/>

<input type=>>radio<> name=>>salario<> value=>>bajo<>> Menos
de 10.000 euros

<br/>

<input type=>>radio<> name=>> salario << value=>>medio<>>
checked> De 10.000 a 20.000 euros

<br/>

<input type=>>radio<> name=>> salario <> value=>>alto<>> Mas
de 20.000 euros

<br/><br/>

Idiomas:

<br/>

<input type=>>checkbox<> name=>>idiomas<> value=>>ingles<>>
checked> Ingles

<br/>

<input type=>>checkbox<> name=>>idiomas<> value=>>frances<>>
Frances

<br/>

<input type=>>checkbox<> name=>>idiomas<> value=>>aleman<>>
Aleman

<br/><br/>

<input type=>>submit<> value=>>Enviar<>/>

</form>

</body>

</html>

```

Como vemos, el rango salarial de un trabajador es algo único, por lo que se programa con botones de opción. Sin embargo, una persona puede hablar más de un idioma, así que esas opciones deben estar programadas con casillas de verificación.

A continuación vamos a ver las distintas opciones y atributos que pueden tomar estos elementos.

- Botones de opción INPUT de tipo RADIO con atributo name.

Este atributo es, si cabe, el más importante cuando se programan botones de radio, ya que todas las opciones que se quieran agrupar en el formulario bajo el mismo grupo de botones tienen que tener el mismo nombre.

Esto quiere decir que, si ponemos dos grupos de botones seguidos, pero cada grupo tiene un nombre distinto, el navegador interpretará que se puede elegir una opción de cada grupo.

Por supuesto, este atributo es esencial a la hora de programar los botones de opciones, ya que si no se utiliza sólo se podrá tener un grupo de botones de radio en todo el formulario.

Veamos un ejemplo de un formulario con una lista de botones de radio con este atributo:

```
<html>
<head>
    <title>Ejemplo de botones de opcion con el atributo name.</title>
</head>
<body>
    <form action=>>mandar_datos.php</> method=>>post</>>
        Nombre:
        <input type=>>text</> name=>>nombre</> value=>>> />
        <br/><br/>
        Edad:

```

## UD1

```

<br/>

<input type=>>radio<> name=>>edad<> value=>>menor<>> Menos
de 18

<br/>

<input type=>>radio<> name=>>edad<> value=>>adulto<>> De 18
a 65

<br/>

<input type=>>radio<> name=>>edad<> value=>>jubilado<>> Mas
de 65

<br/><br/>

<input type=>>submit<> value=>>Enviar<>/>

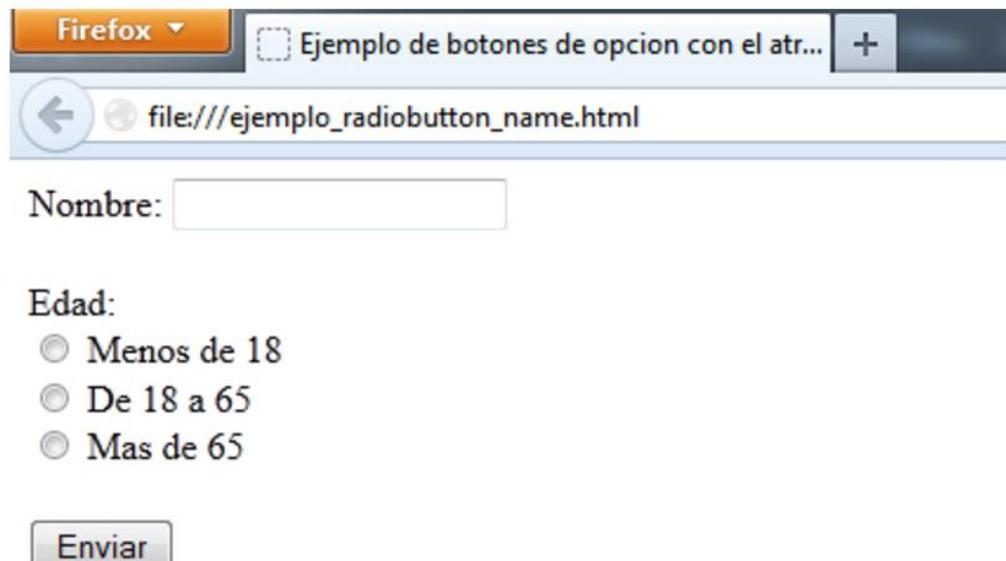
</form>

</body>

</html>

```

Captura de pantalla del resultado que debe dar el código al ejecutarlo en un navegador web.



- Botones de opción INPUT de tipo RADIO con atributo value.

Este atributo es, de cara a la información que va a recibir el servidor, el más relevante, ya que determina cual es la variable que va a recibir dependiendo de lo que marque el usuario.

Esto quiere decir que, aunque el texto de las opciones que se ofrecen sea tremadamente largo, en realidad el servidor sólo recibirá en realidad este valor, que puede ser de un tamaño relativamente pequeño y manejable.

Veamos un ejemplo de un formulario con una lista de botones de radio con este atributo:

```
<html>
<head>
    <title>Ejemplo de botones de opcion con el atributo value.</title>
</head>
<body>
    <form action=>>mandar_datos.php<> method=>>post<>>
        Nombre:
        <input type=>>text<> name=>>nombre<> value=>>> />
        <br/><br/>
        Hora a la que se prefiere almorzar:
        <br/>
        <input type=>>"radio"<> name=>>"almuerzo"<> value=>>"temprano"<>>
        Personalmente prefiero almorzar antes de las una del medio dia.
        <br/>
        <input type=>>"radio"<> name=>>"almuerzo"<> value=>>"normal"<>>
        Personalmente prefiero almorzar entre las una y las tres del medio dia.
        <br/>
```

## UD1

```

<input type="radio" name="almuerzo" value="tarde">
Personalmente prefiero almorzar despues de las tres del
medio dia.

<br/><br/>

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

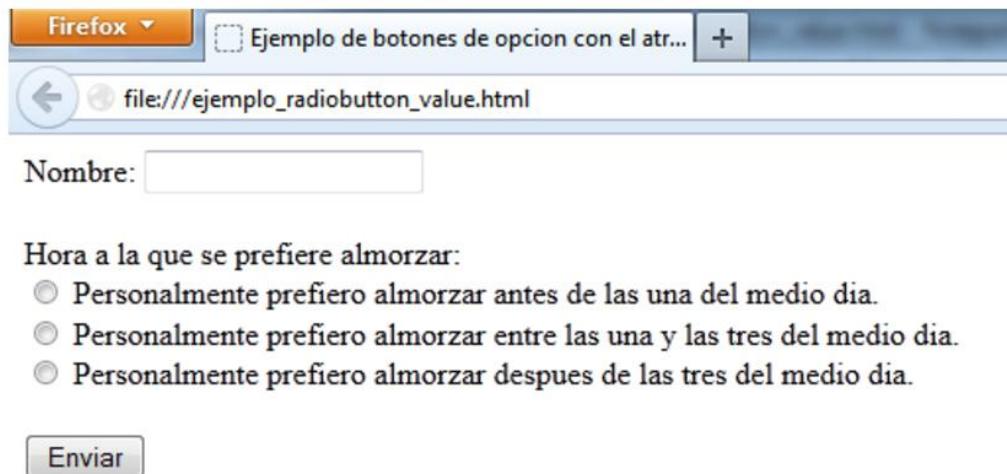
</form>

</body>

</html>

```

Captura de pantalla del resultado que debe dar el código al ejecutarlo en un navegador web.



- Botones de opción INPUT de tipo RADIO con atributo align.

Con este atributo se define como está alineado el botón de opción respecto a los elementos que le rodean. Y el alineamiento puede ser: TOP, MIDDLE, BOTTOM, RIGHT, LEFT, TEXTTOP, BASELINE, ABSMIDDLE, ABSBOTTOM.

Importante: Si no hay elementos alrededor del botón de radio, este atributo no tendrá efecto aparente en el aspecto.

Veamos un ejemplo de un formulario con una lista de botones de radio con este atributo:

```
<html>
```

## UF1304: Elaboración de plantillas y formularios

```

<head>

    <title>Ejemplo de botones de opcion con el atributo
    align.</title>

</head>

<body>

    <form action="mandar_datos.php" method="post">

        Nombre:

        <input type="text" name="nombre" value="" />
        <br/><br/>

        Edad:

        <br/>

        <input type="radio" name="edad" value="menor"
        align="left"> Menos de 18
        <br/>

        <input type="radio" name="edad" value="adulto"
        align="left"> De 18 a 65
        <br/>

        <input type="radio" name="edad" value="jubilado"
        align="left"> Mas de 65
        <br/><br/>

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

    </form>

</body>

</html>

```

Captura de pantalla del resultado que debe dar el código al ejecutarlo en un navegador web.

## UD1

Nombre:

Edad:

- Menos de 18
- De 18 a 65
- Mas de 65

**Enviar**

- Botones de opción INPUT de tipo RADIO con atributo tabindex.

Con este atributo se define el orden de los elementos conforme se vayan accediendo a ellos para obtener el foco mediante la tabulación.

Normalmente el foco se va cogiendo de arriba a abajo, pero con este atributo el programador puede modificar el orden.

Veamos un ejemplo de un formulario con una lista de botones de radio con este atributo:

```
<html>
  <head>
    <title>Ejemplo de botones de opcion con el atributo tabindex.</title>
  </head>
  <body>
    <form action=>>mandar_datos.php</> method=>>post</>>
      Nombre:
      <input type=>>text</> name=>>nombre</> value=>>> />
      <br/><br/>
```

**Edad:**

<br/>

```
<input type=>>radio<> name=>>edad<> value=>>menor<> tabin-
dex=>>2<> Menos de 18
```

<br/>

```
<input type=>>radio<> name=>>edad<> value=>>adulto<> tabin-
dex=>>1<> De 18 a 65
```

<br/>

```
<input type=>>radio<> name=>>edad<> value=>>jubilado<> ta-
bindex=>>3<> Mas de 65
```

<br/><br/>

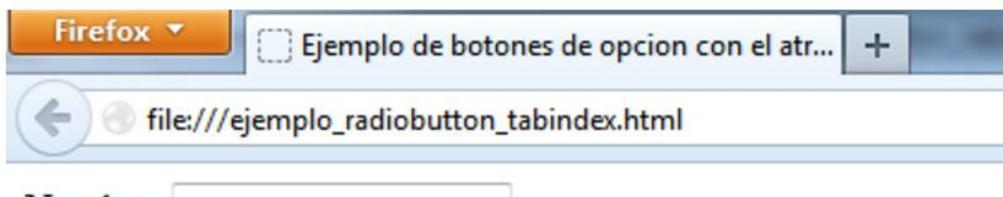
<input type=>>submit<> value=>>Enviar<>/>

</form>

</body>

</html>

Captura de pantalla del resultado que debe dar el código al ejecutarlo en un navegador web.



**Nombre:**

**Edad:**

- Menos de 18
- De 18 a 65
- Mas de 65

**Enviar**

## UD1

- Botones de opción INPUT de tipo RADIO con atributo checked.

Con este atributo se puede asignar un valor como seleccionado nada más carga el formulario, por lo que, si el usuario no selecciona otro valor, este será el que se envíe con el mismo.

Este atributo puede tener múltiples usos. Por ejemplo, si el programador quiere que se introduzca, al menos, una opción, fuerza al usuario a hacerlo seleccionando una automáticamente. Además, normalmente se usará este atributo con la opción que se seleccione más habitualmente, para ahorrarle algo de trabajo al usuario final.

Veamos un ejemplo de un formulario con una lista de botones de radio con este atributo:

```
<html>
<head>
    <title>Ejemplo de botones de opción con el atributo checked.</title>
</head>
<body>
    <form action=>>mandar_datos.php<> method=>>post<>>
        Nombre:
        <input type=>>text<> name=>>nombre<> value=>>> />
        <br/><br/>
        Edad:
        <br/>
        <input type=>>radio<> name=>>edad<> value=>>menor<>> Menos de 18
        <br/>
        <input type=>>radio<> name=>>edad<> value=>>adulto<>> checked> De 18 a 65
        <br/>
```

```

<input type=>radio</> name=>edad</> value=>jubilado</> Mas
de 65

<br/><br/>

<input type=>submit</> value=>Enviar</>

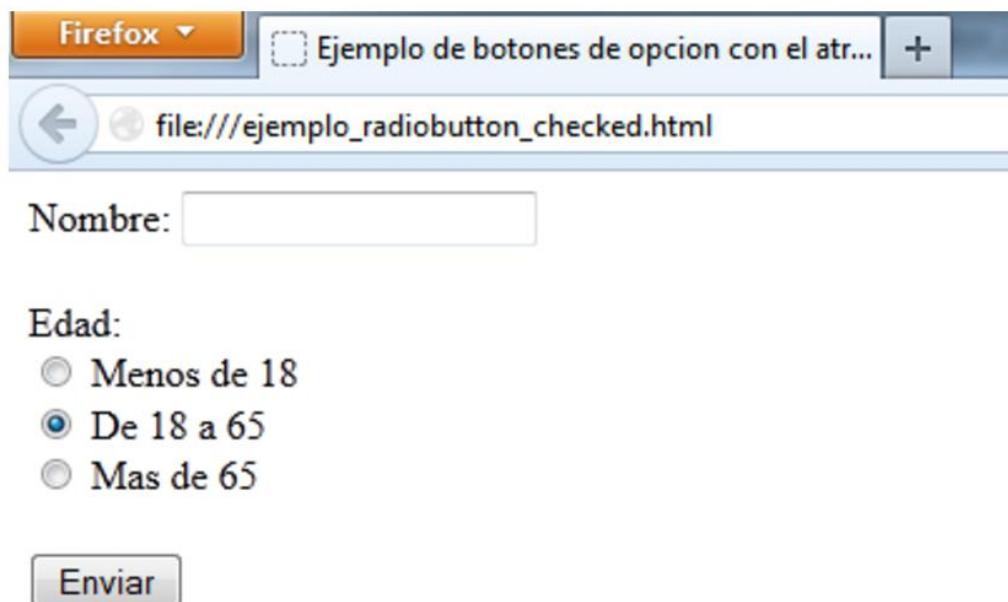
</form>

</body>

</html>

```

Captura de pantalla del resultado que debe dar el código al ejecutarlo en un navegador web.



- Casillas de verificación INPUT de tipo CHECKBOX con atributo name.

Este atributo es, al contrario de lo que pasa en los botones de radio, bastante menos importante que en el otro caso, ya que, como sabemos, en realidad cada una de las opciones que tienen las casillas de verificación, aunque pertenezcan a la misma lista de opciones, son independientes entre sí.

Por supuesto, este atributo sigue siendo muy útil a la hora de asignar a cada una de las opciones un nombre con el que modificarlas conjuntamente, por ejemplo, con un lenguaje de programación como JavaScript.

## UD1

Veamos un ejemplo de un formulario con unas casillas de verificación con este atributo:

```

<html>
  <head>
    <title>Ejemplo de casillas de verificación con el atributo name.</title>
  </head>
  <body>
    <form      action=>>mandar_datos.php</action>      method=>>post</method>
    id=>>formulario</id>

      Nombre:
      <input type=>>text</type> name=>>usuario</name> value=>>> />
      <br/><br/>

      Idiomas:
      <br/>
      <input type=>>checkbox</type> name=>>idiomas</name> value=>>ingles</value>
      Ingles
      <br/>
      <input type=>>checkbox</type> name=>>idiomas</name> value=>>frances</value>
      Frances
      <br/>
      <input type=>>checkbox</type> name=>>idiomas</name> value=>>aleman</value>
      Aleman
      <br/><br/>
      <input type=>>submit</type> value=>>Enviar</value>/>
    </form>
  </body>
</html>

```

Nombre:

**Idiomas:**

Ingles

Frances

Aleman

**Enviar**

Captura de pantalla del resultado que debe dar el código al ejecutarlo en un navegador web.

- Casillas de verificación INPUT de tipo CHECKBOX con atributo value.

Al igual que ocurre con los botones de radio, este atributo es, de cara a la información que va a recibir el servidor, el más relevante, ya que determina cual es la variable que va a recibir dependiendo de lo que marque el usuario.

Lo que viene a decir que, aunque el texto de las opciones que se ofrecen sea tremadamente largo, en realidad el servidor sólo recibirá en realidad este valor, que puede ser de un tamaño relativamente pequeño y manejable.

Veamos un ejemplo de un formulario con unas casillas de verificación con este atributo:

```
<html>
<head>
    <title>Ejemplo de casillas de verificación con el atributo value.</title>
</head>
```

## UD1

```

<body>

    <form      action=>>mandar_datos.php</>      method=>>post</>
    id=>>formulario<>>

        Nombre:

        <input type=>>text</> name=>>usuario</> value=>>> />

        <br/><br/>

        Consumos habituales:

        <br/>

        <input type=>>checkbox</> name=>>opinion</> value=>>lacteo</>
        Consumo regularmente muchos productos lacteos

        <br/>

        <input type=>>checkbox</> name=>>opinion</> value=>>fruta</>
        Consumo regularmente mucha fruta y verdura

        <br/>

        <input type=>>checkbox</> name=>>opinion</> value=>>carne</>
        Consumo regularmente mucha carne y pescado

        <br/><br/>

        <input type=>>submit</> value=>>Enviar</>/>

    </form>

</body>

</html>

```

Captura de pantalla del resultado que debe dar el código al ejecutarlo en un navegador web.

Nombre:

**Consumos habituales:**

- Consumo regularmente muchos productos lácteos
- Consumo regularmente mucha fruta y verdura
- Consumo regularmente mucha carne y pescado

**Enviar**

- Casillas de verificación INPUT de tipo CHECKBOX con atributo align.

De nuevo, nos encontramos con un atributo similar al de los botones de radio. Con este atributo se define como está alineada la casilla de verificación respecto a los elementos que le rodean. Y el alineamiento puede ser: TOP, MIDDLE, BOTTOM, RIGHT, LEFT, TEXTTOP, BASELINE, ABS-MIDDLE, ABSBOTTOM.



Si no hay elementos alrededor del checkbox, este atributo no tendrá efecto aparente en el aspecto.

Veamos un ejemplo de un formulario con unas casillas de verificación con este atributo:

```
<html>
<head>
    <title>Ejemplo de casillas de verificación con el atributo align.</title>
```

## UD1

```

</head>

<body>

    <form      action="mandar_datos.php"      method="post"
id="formulario">

        Nombre:

        <input type="text" name="usuario" value="" />
        <br/><br/>

        Idiomas:

        <br/>

        <input type="checkbox" name="idiomas" value="ingles"
align=left"> Ingles
        <br/>

        <input type="checkbox" name="idiomas" value="frances"
align=left"> Frances
        <br/>

        <input type="checkbox" name="idiomas" value="aleman"
align=left"> Aleman
        <br/><br/>

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

    </form>

</body>

</html>

```

Captura de pantalla del resultado que debe dar el código al ejecutarlo en un navegador web.

- Casillas de verificación INPUT de tipo CHECKBOX con atributo tabindex.

Con este atributo se define el orden de los elementos conforme se vayan accediendo a ellos para obtener el foco mediante la tabulación.

Normalmente el foco se va cogiendo de arriba a abajo, pero con este atributo el programador puede modificar el orden.

Veamos un ejemplo de un formulario con unas casillas de verificación con este atributo:

```
<html>

<head>

    <title>Ejemplo de casillas de verificación con el atributo tabindex.</title>

</head>

<body>

    <form      action=>>mandar_datos.php<>      method=>>post<>
    id=>>formulario<>>

        Nombre:

        <input type=>>text<> name=>>usuario<> value=>>> />

        <br/><br/>

        Idiomas:

        <br/>

        <input type=>>checkbox<> name=>>idiomas<> value=>>ingles<>
        tabindex=>>2<>> Ingles

        <br/>

        <input type=>>checkbox<> name=>>idiomas<> value=>>frances<>
        tabindex=>>1<>> Frances

        <br/>

        <input type=>>checkbox<> name=>>idiomas<> value=>>aleman<>
        tabindex=>>3<>> Aleman

        <br/><br/>

        <input type=>>submit<> value=>>Enviar<>/>

    </form>

</body>

</html>
```

## UD1

Captura de pantalla del resultado que debe dar el código al ejecutarlo en un navegador web.

The screenshot shows a Firefox browser window with the title bar "Ejemplo de casillas de verificacion con el...". The address bar displays "file:///ejemplo\_checkbox\_tabindex.html". The page content includes:

- A text input field labeled "Nombre:" followed by an empty input box.
- A section labeled "Idiomas:" with three checkboxes:
  - Inglés
  - Francés
  - Alemán
- A large "Enviar" button.

- Casillas de verificación INPUT de tipo CHECKBOX con atributo checked.

Con este atributo se puede asignar un valor como seleccionado nada más carga el formulario, por lo que, si el usuario no selecciona otro valor, este será el que se envíe con el mismo.

En el caso de las casillas de verificación, efectivamente puede haber más de un elemento seleccionado por defecto cuando cargue el formulario en el navegador, sin embargo, al igual que pasa en los botones de opciones, es recomendable siempre dejar seleccionado de antemano los valores que más se van a seleccionar en el formulario para facilitarle el trabajo al usuario final.

Veamos un ejemplo de un formulario con unas casillas de verificación con este atributo:

```
<html>
<head>
  <title>Ejemplo de casillas de verificacion con el atributo checked.</title>
```

```

</head>

<body>

    <form      action=>>mandar_datos.php<>      method=>>post<>
    id=>>formulario<>>

        Nombre:

        <input type=>>text<> name=>>usuario<> value=>>> />

        <br/><br/>

        Idiomas:

        <br/>

        <input type=>>checkbox<> name=>>idiomas<> value=>>ingles<>
        checked> Ingles

        <br/>

        <input type=>>checkbox<> name=>>idiomas<> value=>>frances<>
        Frances

        <br/>

        <input type=>>checkbox<> name=>>idiomas<> value=>>aleman<>
        checked> Aleman

        <br/><br/>

        <input type=>>submit<> value=>>Enviar<>/>

    </form>

</body>

</html>

```

Captura de pantalla del resultado que debe dar el código al ejecutarlo en un navegador web.

## UD1

Nombre:

**Idiomas:**

- Ingles
- Frances
- Alemán

**Enviar**

En este momento el alumno deberá poder saber las diferencias entre las casillas de verificación y los botones de opciones, así como todos sus atributos y cuándo debe usar uno u otro cuando programe un formulario.

A continuación vemos un ejemplo con ambas opciones con todos sus atributos:

```
<html>
<head>
    <title>Ejemplo de casillas de verificacion y botones de opcion.</title>
</head>
<body>
    <form      action=>>mandar_opinion.php<>      method=>>post<>
id=>>formulario<>>
        Encuesta de opinion.

        <br/><br/>

        Que le ha parecido la encuesta:

        <br/>
```

```

<input type="radio" name="opinion" value="buena" tabindex="2" align="left"> Me ha parecido buena en general
<br/>

<input type="radio" name="opinion" value="regular" tabindex="1" align="left" checked> No creo que haya sido ni buena ni mala
<br/>

<input type="radio" name="opinion" value="mala" tabindex="3" align="left"> Creo que ha sido una mala encuesta
<br/><br/>

Como ha conocido nuestra empresa:

<br/>

<input type="checkbox" name="fuente" value="internet" tabindex="3" align="left" checked> A traves de anuncios en internet
<br/>

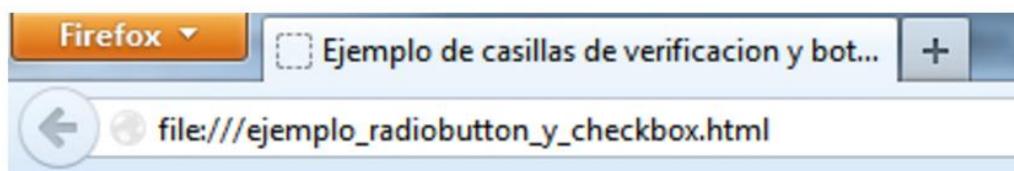
<input type="checkbox" name="fuente" value="televisión" tabindex="2" align="left"> A traves de anuncios en televisión
<br/>

<input type="checkbox" name="fuente" value="prensa" tabindex="1" align="left"> A traves de anuncios en la prensa escrita
<br/><br/>

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

Captura de pantalla del resultado que debe dar el código al ejecutarlo en un navegador web.

## UD1

**Encuesta de opinion.**

**Que le ha parecido la encuesta:**

- Me ha parecido buena en general
- No creo que haya sido ni buena ni mala
- Creo que ha sido una mala encuesta

**Como ha conocido nuestra empresa:**

- A traves de anuncios en internet
- A traves de anuncios en television
- A traves de anuncios en la prensa escrita

**Enviar**

### 1.3.5. Utilización de campos de texto

Los campos de texto son controles o elementos del formulario donde los usuarios pueden introducir texto para llenar los requerimientos del mismo, ya sea un texto más corto o más largo, o incluso un texto cifrado en la lectura.

Estos elementos son los más comunes de los formularios, ya que al fin y al cabo son los que permiten que el usuario interactúe de forma total, como por ejemplo ocurre en los formularios de los buscadores, en los de las tiendas online, o simplemente en formularios para llenar datos personales o de registro en páginas web.

Hay tres tipos principales de campos de texto:

- Los cuadros de texto.

- Los cuadros de contraseña.
- Las áreas de texto.

Los cuadros de texto son los controles más utilizados en los formularios, y a la vez los más simples de cara al usuario, que no del programador.

Se incluyen en los formularios usando la etiqueta INPUT y el tipo TEXT. Y por supuesto, tienen múltiples atributos importantes:

- Size.
- Maxlength.
- Name.
- Value.
- Align.
- Tabindex.

Todos estos los veremos en profundidad a continuación, con unas líneas de código para cada uno de ellos.

Pero, si queremos ver el ejemplo de cuadro de texto probablemente más utilizado del mundo, sólo hay que recurrir a Google:

Captura de pantalla del buscador de Google en el navegador Mozilla Firefox.



## UD1

En esta captura de pantalla podemos observar el formulario que, como ya explicamos al principio de esta Unidad Didáctica, es probablemente el más reconocible pero a su vez el más desconocido en general del mundo.

En este formulario podemos ver, además de los dos botones que ya sabemos su funcionamiento por los puntos anteriores, un gran cuadro de texto donde el usuario introduce lo que quiere buscar.

Un elemento simple, y a la vez efectivo.

Los cuadros de contraseña son familia directa de los cuadros de texto, pero con una diferencia principal, lo que se escribe en ellos no se ve a simple vista. Lo cual los convierte en los elementos ideales para que el usuario pueda introducir su contraseña con la seguridad de que está a salvo de las miradas indiscretas de personas que vean la pantalla mientras la introduce.

Los distintos navegadores ocultan, de forma personalizada, los caracteres según los va escribiendo el usuario. El método más antiguo es utilizar asteriscos, pero hoy en día se pueden ver todo tipo de sistemas, como por ejemplo hacerlo con círculos.

Una gran pega de este cuadro de contraseñas es que, si el usuario comete un error, no puede ver que es lo que ha escrito y rectificarlo. Pero las ventajas que se ganan respecto a la seguridad hacen que este problema se pueda obviar normalmente.

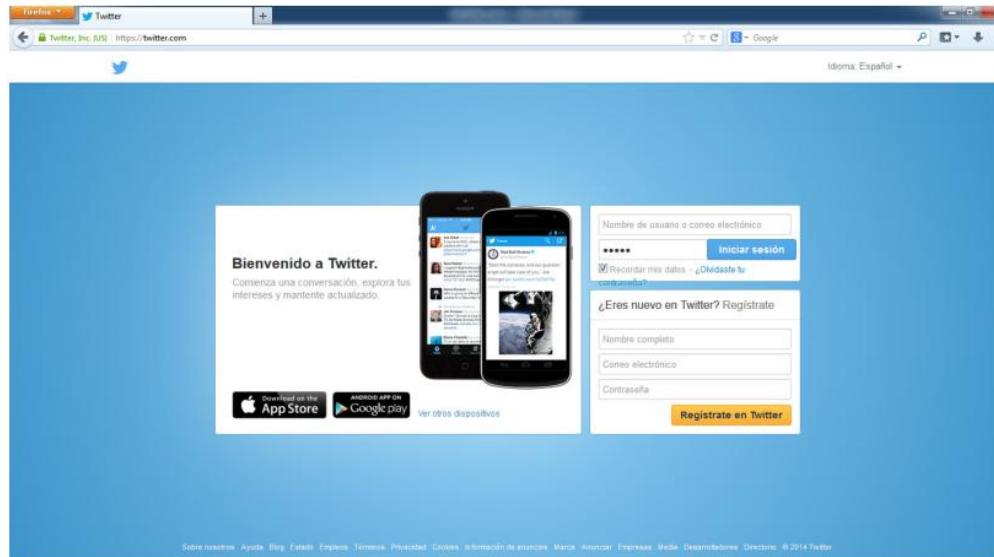
La forma de incluirlos en los formularios es usando la etiqueta INPUT y el tipo PASSWORD. Y por supuesto, tienen múltiples atributos importantes:

- Size.
- Maxlength.
- Name.
- Value.
- Align.
- Tabindex.

Al igual que con los cuadros de texto, todos estos los veremos en profundidad a continuación, con unos programas creados para cada uno de ellos.

A continuación podemos ver un ejemplo de cuadro de contraseña, en este caso ocultos los caracteres por círculos, en la página principal de Twitter:

Captura de pantalla de la página principal de Twitter en el navegador Mozilla Firefox.



Se puede decir que prácticamente todas las páginas del mundo que tengan un formulario de registro o de ingreso en su página cuentan con, al menos, un cuadro de contraseña. Lo que hace que este sea, especialmente relevante.

Y por último, están las áreas de texto. Estas son como grandes cuadros de texto pero con varias líneas y la posibilidad de escribir o introducir grandes cantidades de palabras.

Este elemento es menos utilizado que los otros, pero no por ello deja de ser importante, ya que a veces en los formularios hay que introducir en un lugar muchas frases o incluso párrafos, y un cuadro de texto se queda muy corto, debido a su límite máximo de caracteres, para este fin.

La forma de incluirlos en los formularios es también distinta a los otros dos elementos, ya que no se usa la etiqueta INPUT, si no la etiqueta TEXTAREA, la cual además actúa distinta, ya que necesita ser cerrada con una etiqueta de cierre, y el texto contenido en su interior cuando carga el formulario se debe introducir entre esas dos etiquetas.

Las áreas de texto también tienen múltiples atributos importantes, como son:

- Rows.

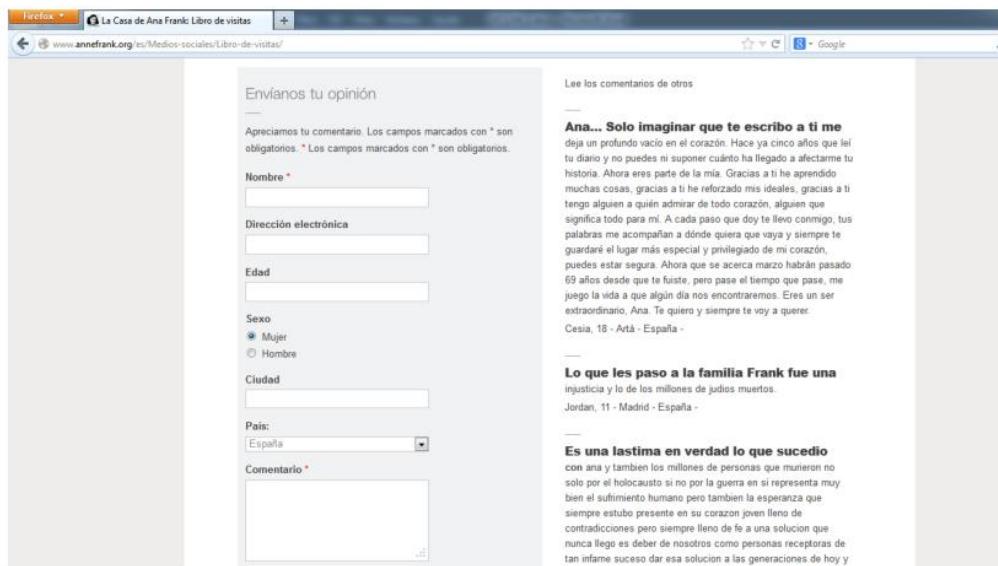
## UD1

- Cols.
- Name.
- Tabindex.
- Wrap.

Y por supuesto, al igual que ocurre con los cuadros de texto y los cuadros de contraseña, todos estos atributos los veremos en profundidad a continuación, con unos programas creados para cada uno de ellos.

Este elemento de campo de texto es especialmente utilizado en formularios donde hay sección de comentarios, o en los libros de visitas, los cuales cada vez se encuentran más en desuso. Por ejemplo, podemos encontrar un área de texto en el libro de visitas de la página web en memoria de Ana Frank:

Captura de pantalla del libro de visitas de la página web en memoria de Ana Frank en el navegador Mozilla Firefox.



Y ya sólo queda ver un ejemplo del código, antes de entrar en profundidad en los atributos de cada uno de los controles:

```
<html>
<head>
```

```

<title>Ejemplo de campos de texto.</title>

</head>

<body>

<form action="campos_de_texto.php" method="post">

<input type="text" name="campo_texto" value="Campo de
Texto" />

<br/>

<input type="password" name="campo_password"
value="Password" />

<br/>

<textarea name="area_de_texto" cols="40" rows="5"
>Area de Texto</textarea>

</form>

</body>

</html>

```

- Cuadros de texto INPUT de tipo TEXT con atributo size.

Los cuadros de texto tienen una altura específica que siempre es de una línea, al contrario que le ocurre a las áreas de texto. Sin embargo, su anchura puede variar en el número de caracteres usando el atributo SIZE.

Este atributo no es esencial en la programación de este tipo de elemento, ya que si no se incluye el cuadro de texto tendrá la anchura por defecto, que es de veinte caracteres. Sin embargo, para un diseño eficaz siempre será necesario modificarla.

Por ejemplo, no es lo mismo la anchura que debe tener un campo de texto especialmente diseñado para introducir los nombres y apellidos de una persona, o su dirección, que un cuadro de texto diseñado para introducir la edad, o un año concreto. Obviamente en el último caso la anchura debe ser mucho mejor que en el primero.

Veamos un ejemplo de un formulario con un cuadro de texto con este atributo:

```
<html>
```

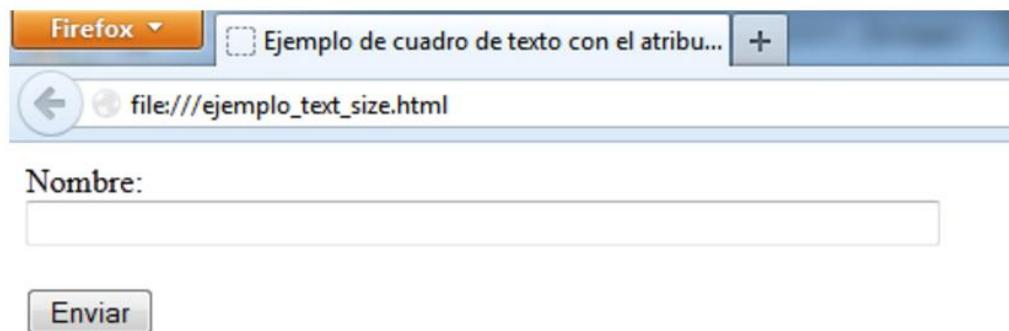
## UD1

```

<head>
    <title>Ejemplo de cuadro de texto con el atributo
size.</title>
</head>
<body>
    <form action="mandar_datos.php" method="post">
        Nombre:
        <br/>
        <input type="text" name="nombre" size="70" />
        <br/>
        <br/>
        <input type="submit" value="Enviar"/>
    </form>
</body>
</html>

```

Captura de pantalla del resultado que debe dar el código al ejecutarlo en un navegador web.



Véase que la anchura del cuadro de texto es mucho mayor de la habitual.

- Cuadros de texto INPUT de tipo TEXT con atributo maxlength.

En los cuadros de texto además de poder variar en el número de caracteres usando el atributo SIZE, se puede asignar el número máximo de caracteres que se pueden meter en el mismo usando el atributo MAXLENGTH.

Este atributo, al igual que el de SIZE, no es esencial en la programación de este tipo de elemento, ya que si no se incluye el cuadro de texto no tendrá límite de caracteres por defecto. Sin embargo, para un diseño funcional lógico, y para un acotamiento del rango de resultados posibles, siempre será necesario implementarlo.

Por ejemplo, no es lo mismo el rango máximo de caracteres para introducir un año o la edad, que el que permita una dirección. Obviamente en el último caso el número de caracteres debe ser mucho mayor que en el anterior.

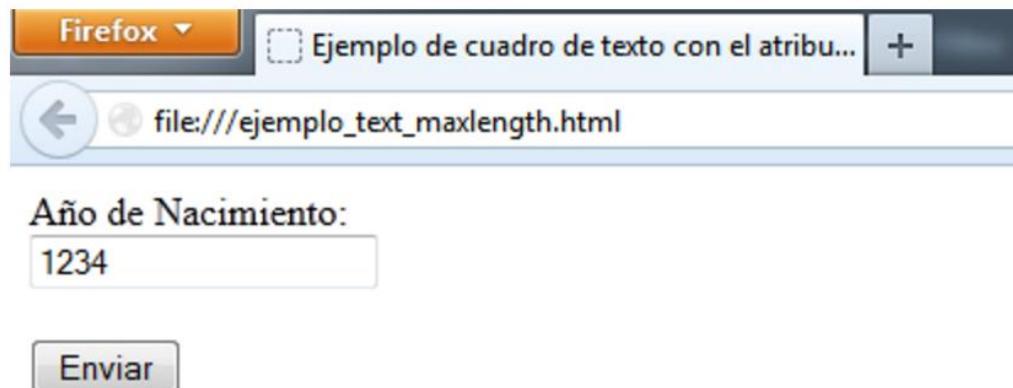
Veamos un ejemplo de un formulario con un cuadro de texto con este atributo:

```
<html>
<head>
    <title>Ejemplo de cuadro de texto con el atributo
    maxlength.</title>
</head>
<body>
    <form action=>mandar_nacimiento.php</action>
        A&ntilde;o de Nacimiento:
        <br/>
        <input type="text" name="nombre" maxlength="4" />
        <br/>
        <br/>
        <input type="submit" value="Enviar"/>
    </form>
```

## UD1

```
</body>
</html>
```

Captura de pantalla del resultado que debe dar el código al ejecutarlo en un navegador web.



En ese cuadro de texto no podrán introducirse más de 4 caracteres.

- Cuadros de texto INPUT de tipo TEXT con atributo name.

Los cuadros de texto también pueden ser referenciados por otros lenguajes de script, especialmente por JavaScript. Para que ocurra esto, se puede utilizar el atributo ID, pero, como el alumno ya debe conocer, también existe un atributo propio de HTML que cumple esta función y que además puede ser asignado a varios elementos, el atributo NAME.

Además, en el caso de los formularios, y en los cuadros de texto en concreto, este atributo cobra mayor importancia si cabe, ya que sólo aquellos cuadros de texto con un atributo name pasarán sus valores al servidor cuando se envíen los datos del formulario.

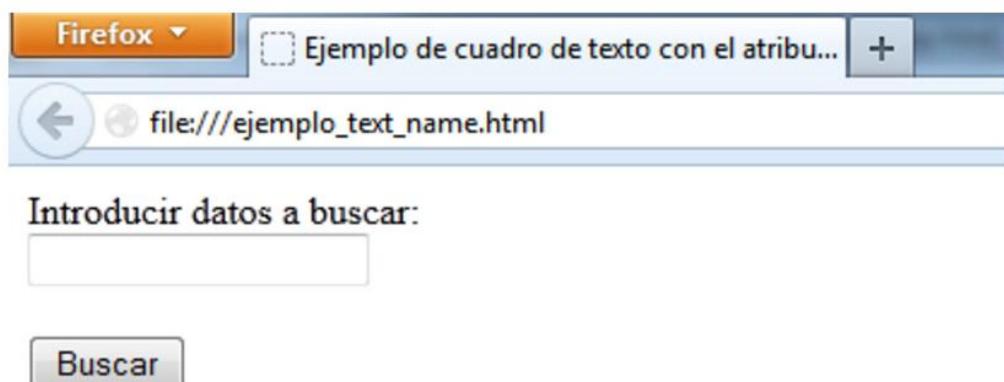
Esto quiere decir que si un campo de texto no tiene un atributo name asignado, aunque el usuario final lo rellene, no se enviará su información al servidor aunque se pulse un botón de submit.

Veamos un ejemplo de un formulario con un cuadro de texto con este atributo:

```
<html>
<head>
```

```
<title>Ejemplo de cuadro de texto con el atributo  
name.</title>  
  
</head>  
  
<body>  
  
<form action=>>buscador.php<> method=>>post<>>  
  
Introducir datos a buscar:  
  
<br/>  
  
<input type="text" name="buscador" />  
  
<br/>  
  
<br/>  
  
<input type="submit" value="Buscar"/>  
  
</form>  
  
</body>  
  
</html>
```

Captura de pantalla del resultado que debe dar el código al ejecutarlo en un navegador web.



Véase que si el cuadro de texto no tuviese el atributo name aunque se le diese al botón de submit no se enviaría nada.

## UD1

- Cuadros de texto INPUT de tipo TEXT con atributo value.

Los cuadros de texto están diseñados y pensados para que el usuario final introduzca los datos, de forma interactiva, para que cuando se envíe el formulario estos datos sean enviados al servidor para su procesamiento.

Pero hay veces que el programador necesita que un cuadro de texto esté relleno con algunos caracteres cuando se carga el formulario, ya sea por una intención funcional, por ejemplo introduciendo los datos más comunes que suelen ser llenados en ese campo, o por una intención estética, como por ejemplo introduciendo el texto indicando al usuario final lo que debe llenarse en el interior del mismo, en lugar de antes del cuadro de texto.

En todo caso, para hacer esta función hay que usar el atributo VALUE, el cual mostrará su contenido en el cuadro de texto al cargar el formulario.

Veamos un ejemplo de un formulario con un cuadro de texto con este atributo:

```
<html>
<head>
    <title>Ejemplo de cuadro de texto con el atributo value.</title>
</head>
<body>
    <form action=>enviar_datos.php</action> method=>post</method>
        <input type=>text</type> name=>nombre</name> value=>Nombre</value>
        <br/>
        <input type=>text</type> name=>apellidos</name> value=>Apellidos</value>
        <br/>
        <input type=>text</type> name=>direccion</name> value=>Direccion</value>
        <br/>
```

```

<br/>

<input type=>submit</> value=>Enviar</>

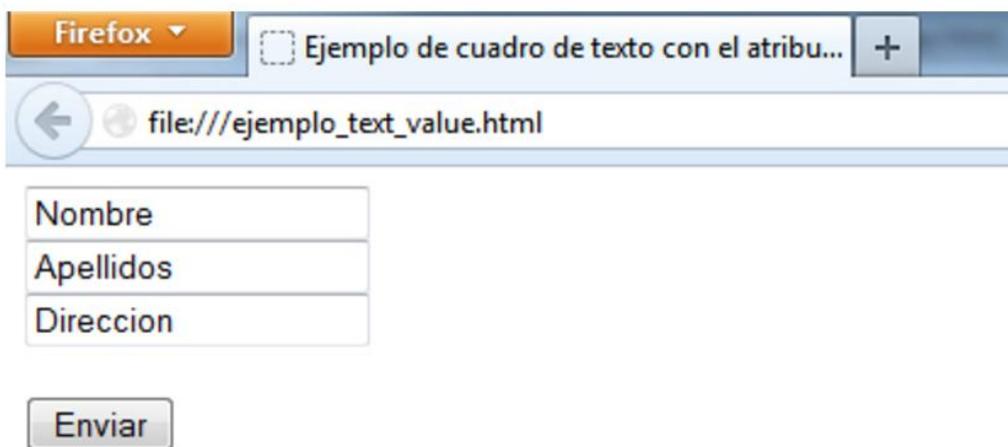
</form>

</body>

</html>

```

Captura de pantalla del resultado que debe dar el código al ejecutarlo en un navegador web.



En los cuadros de texto se le informa al usuario lo que tiene que rellenar en cada uno.

- Cuadros de texto INPUT de tipo TEXT con atributo align.

Los cuadros de texto, al igual que el resto de elementos de un formulario, pueden encontrarse junto con otros elementos de diseño, especialmente imágenes, que descoloren su maquetación.

Para ello nos encontramos con el atributo ALIGN, que define como está alineado el cuadro de texto respecto a los elementos que le rodean.

El alineamiento puede ser: TOP, MIDDLE, BOTTOM, RIGHT, LEFT, TEXTTOP, BASELINE, ABSMIDDLE, ABSBOTTOM.

## UD1



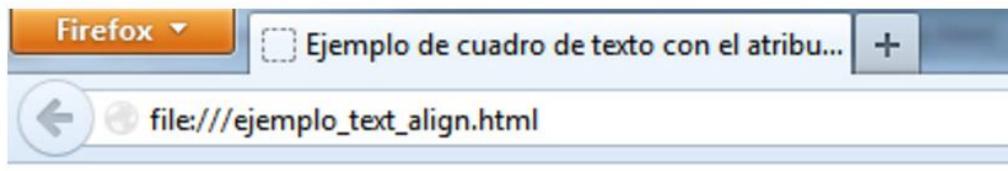
Si no hay elementos alrededor del cuadro de texto, este atributo no tendrá efecto aparente en el aspecto.

---

Veamos un ejemplo de un formulario con un cuadro de texto con este atributo:

```
<html>
  <head>
    <title>Ejemplo de cuadro de texto con el atributo align.</title>
  </head>
  <body>
    <form action="buscador.php" method="post">
      Introducir datos a buscar:
      <br/>
      <input type="text" name="buscador" align="left"/>
      <br/>
      <br/>
      <input type="submit" value="Buscar"/>
    </form>
  </body>
</html>
```

Captura de pantalla del resultado que debe dar el código al ejecutarlo en un navegador web.



### Introducir datos a buscar:

**Buscar**

Véase que para que actúe el alineamiento debe estar rodeado de elementos como imágenes.

- Cuadros de texto INPUT de tipo TEXT con atributo tabindex.

Los cuadros de texto, como el resto de elementos de un formulario o una web en general programada en HTML, obtienen el foco según se van dando pulsaciones a la tecla tabulador según el orden en el que han sido dispuestas. Esto es muy útil para determinar, normalmente, el orden en el que se quieren incluir los elementos en la página, pero a veces es necesario incluir un orden visual y, a su vez, cambiar el orden de obtención de foco.

Para hacer esto tenemos el atributo TABINDEX, el cual define el orden de los elementos conforme se vayan accediendo a ellos para obtener el foco mediante la tabulación.



El resto de elementos del formulario que no tengan definidos este atributo conservarán el orden de tabulación normal.

---

Veamos un ejemplo de un formulario con un cuadro de texto con este atributo:

```
<html>
```

## UD1

```
<head>

    <title>Ejemplo de cuadro de texto con el atributo ta-
    bindex.</title>

</head>

<body>

    <form action=>>enviar_datos.php<> method=>>post<>>

        <input type=>>text<> name=>>nombre<> value=>>Nombre<> tabin-
        dex=>>2<> />

        <br/>

        <input type=>>text<> name=>>apellidos<> value=>>Apellidos<>
        tabindex=>>3<> />

        <br/>

        <input type=>>text<> name=>>direccion<> value=>>Direccion<>
        tabindex=>>1<> />

        <br/>

        <br/>

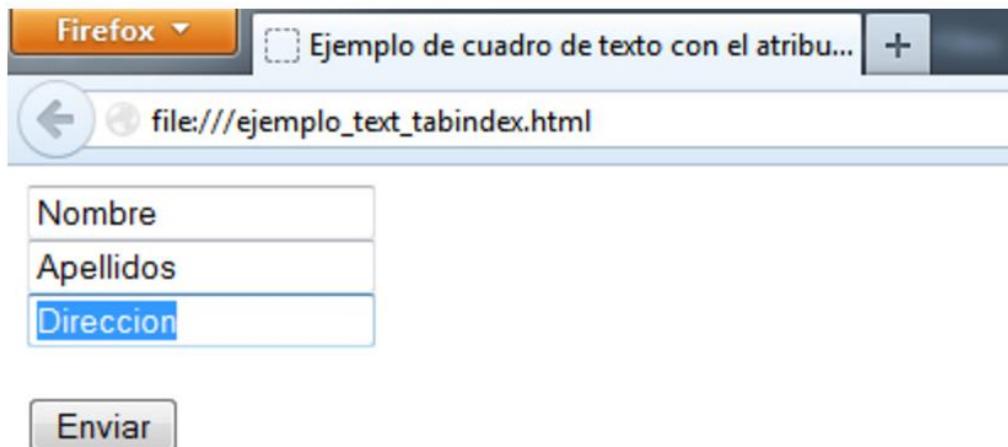
        <input type=>>submit<> value=>>Enviar<>/>

    </form>

</body>

</html>
```

Captura de pantalla del resultado que debe dar el código al ejecutarlo en un navegador web.



- Cuadros de contraseña INPUT de tipo PASSWORD con atributo size.

Los cuadros de contraseña, al igual que los cuadros de texto, también tienen una altura específica, que siempre es de una línea, al contrario que le ocurre a las áreas de texto. Sin embargo, su anchura puede variar en el número de caracteres usando el atributo SIZE.

Este atributo no es esencial en la programación de este tipo de elemento, ya que si no se incluye el cuadro de texto tendrá la anchura por defecto, que es de veinte caracteres. Sin embargo, para un diseño eficaz siempre será necesario modificarla, especialmente en las contraseñas, que normalmente tienen un rango de caracteres aún más ajustados que los valores que se suelen introducir normalmente en los cuadros de texto.

Por ejemplo, no es lo mismo la anchura que debe tener un campo de texto para introducir una dirección, que debe ser muy largo, que el usado para introducir una contraseña, que salvo excepciones suele tener un tamaño mucho más reducido.

Veamos un ejemplo de un formulario con un cuadro de contraseña con este atributo:

```
<html>
<head>
    <title>Ejemplo de cuadro de contraseña con el atributo size.</title>
</head>
```

## UD1

```
<body>

    <form action=>ingresar.php</action> method=>post</method>

        Usuario:

        <input type=>text</type> name=>nombre</name> size=>40</size> />

        <br/>

        Password:

        <input type=>password</type> name=>password</name> size=>10</size> />

        <br/>

        <br/>

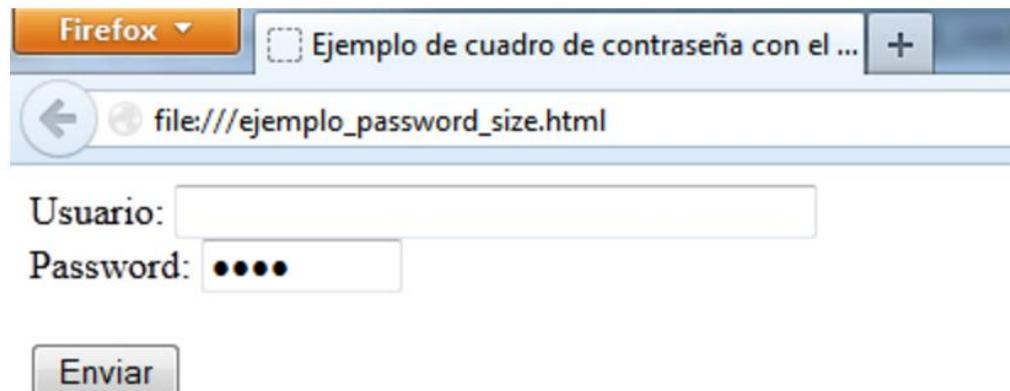
        <input type=>submit</type> value=>Enviar</value>/>

    </form>

</body>

</html>
```

Captura de pantalla del resultado que debe dar el código al ejecutarlo en un navegador web.



- Cuadros de contraseña INPUT de tipo PASSWORD con atributo maxlength.

En los cuadros de contraseña el programador también tiene la opción, además de poder variar en el número de caracteres usando el atributo SIZE, de asignar el número máximo de caracteres que se pueden introducir en el mismo usando el atributo MAXLENGTH.

Este atributo, al igual que el de SIZE, no es esencial en la programación de este tipo de elemento, ya que si no se incluye el cuadro de texto no tendrá límite de carácteres por defecto. Sin embargo, para los cuadros de contraseña sí que suele ser bastante más relevante que en los cuadros de texto, ya que las contraseñas suelen tener un número máximo de caracteres, así que acotarlo siempre suele ser una buena idea.

Por ejemplo, si una contraseña debe estar comprendida entre cuatro y doce caracteres, lo más lógico es que el MAXLENGTH de ese campo sea de doce, ya que no tiene sentido que el usuario final pueda introducir más caracteres en ese campo.

Veamos un ejemplo de un formulario con un cuadro de contraseña con este atributo:

```
<html>
<head>
    <title>Ejemplo de cuadro de contraseña con el atributo maxlength.</title>
</head>
<body>
    <form action=>>ingresar.php<> method=>>post<>>
        Usuario:
        <input type=>>text<> name=>>nombre<> />
        <br/>
        Password:
        <input type=>>password<> name=>>password<> maxlength=>>6<>
    />
    <br/>
```

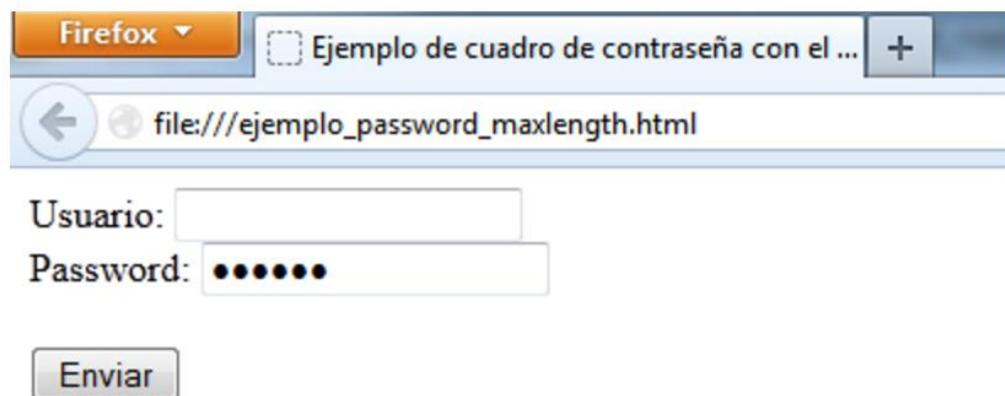
## UD1

```

<br/>
<input type=>submit</input value=>Enviar</input>
</form>
</body>
</html>

```

Captura de pantalla del resultado que debe dar el código al ejecutarlo en un navegador web.



Véase que no se pueden introducir más caracteres en el cuadro de contraseña.

- Cuadros de contraseña INPUT de tipo PASSWORD con atributo name.

Los cuadros de contraseña, al igual que los cuadros de texto, también pueden ser referenciados por otros lenguajes de script como JavaScript. Para que ocurra esto, como ya hemos visto, se puede utilizar el atributo ID, pero también existe el atributo NAME, un atributo propio de HTML que cumple esta función y que además puede ser asignado a varios elementos.

Y sin olvidar que en el caso de los formularios, y en los cuadros de contraseña en concreto, este atributo cobra mayor importancia si cabe, ya que sólo aquellos cuadros de contraseña con un atributo name pasarán sus valores al servidor cuando se envíen los datos del formulario.

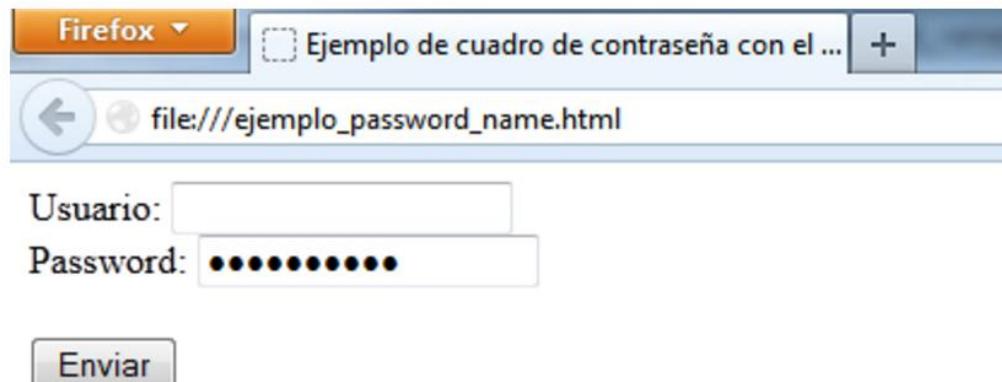
Esto quiere decir que si un campo de contraseña no tiene un atributo name asignado, aunque el usuario final lo rellene, no se enviará su información al servidor aunque se pulse un botón con la función de submit.

Veamos un ejemplo de un formulario con un cuadro de contraseña con este atributo:

```
<html>
<head>
    <title>Ejemplo de cuadro de contraseña con el atributo name.</title>
</head>
<body>
    <form action=>ingresar.php</action> method=>post</method>
        Usuario:
        <input type=>text</type> name=>nombree</name> />
        <br/>
        Password:
        <input type=>password</type> name=>password</name> />
        <br/>
        <br/>
        <input type=>submit</type> value=>Enviar</value> />
    </form>
</body>
</html>
```

Captura de pantalla del resultado que debe dar el código al ejecutarlo en un navegador web.

## UD1



- Cuadros de contraseña INPUT de tipo PASSWORD con atributo value.

Los cuadros de contraseña, al igual que los cuadros de texto, están diseñados y pensados para que el usuario final introduzca los datos, de forma interactiva, para que cuando se envíe el formulario estos datos sean enviados al servidor para su procesamiento. En concreto este campo está diseñado especialmente para el envío de contraseñas.

Pero hay veces que el programador necesita que un cuadro de contraseñas esté relleno con algunos caracteres cuando se carga el formulario. Normalmente esto se hace por una mera utilidad funcional, ya que muchas veces el usuario ya ha introducido anteriormente su contraseña, por lo que se pasa una variable al atributo value con el mismo valor de la anteriormente introducida por el usuario, y así, si este lo desea, no tiene que volver a introducirla cada vez que entra en ese formulario.

Para hacer esta función hay que usar, como ya se ha dicho, el atributo VALUE, el cual llenará el contenido en el cuadro de contraseña al cargar el formulario.

Veamos un ejemplo de un formulario con un cuadro de contraseña con este atributo:

```
<html>
<head>
    <title>Ejemplo de cuadro de contraseña con el atributo value.</title>
</head>
<body>
```

```

<form action=>ingresar.php</action method=>post</>

    Usuario:
    <input type=>text</input name=>nombree />
    <br/>

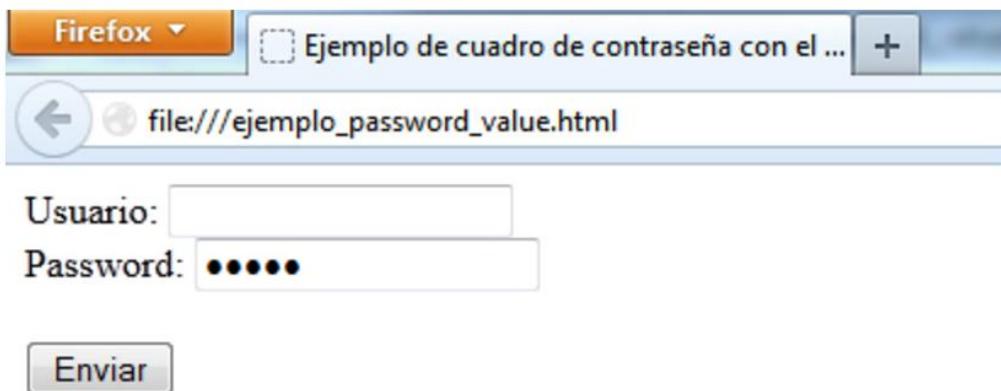
    Password:
    <input type=>password</input name=>password value=>12345</>
    <br/>
    <br/>
    <input type=>submit</input value=>Enviar</>
</form>

</body>

</html>

```

Captura de pantalla del resultado que debe dar el código al ejecutarlo en un navegador web.



- Cuadros de contraseña INPUT de tipo PASSWORD con atributo align.

Los cuadros de contraseña, al igual que el resto de elementos de un formulario, pueden encontrarse junto con otros elementos de diseño, especialmente imágenes, que descoloquen su maquetación.

## UD1

Para ello nos encontramos con el atributo ALIGN, que define como está alineado el cuadro de texto respecto a los elementos que le rodean.

El alineamiento puede ser: TOP, MIDDLE, BOTTOM, RIGHT, LEFT, TEXTTOP, BASELINE, ABSMIDDLE, ABSBOTTOM.



Si no hay elementos alrededor del cuadro de contraseña, este atributo no tendrá efecto aparente en el aspecto.

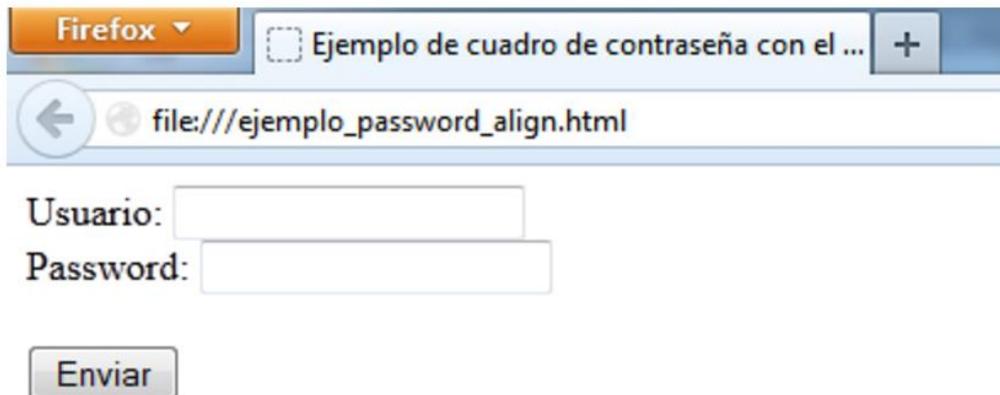
---

Veamos un ejemplo de un formulario con un cuadro de contraseña con este atributo:

```
<html>
<head>
    <title>Ejemplo de cuadro de contraseña con el atributo align.</title>
</head>
<body>
    <form action="ingresar.php" method="post">
        Usuario:
        <input type="text" name="nombre" />
        <br/>
        Password:
        <input type="password" name="password" align="left" />
        <br/>
        <br/>
        <input type="submit" value="Enviar"/>
    </form>
```

```
</body>
</html>
```

Captura de pantalla del resultado que debe dar el código al ejecutarlo en un navegador web.



Véase que para que actúe el alineamiento debe estar rodeado de elementos como imágenes.

- Cuadros de contraseña INPUT de tipo PASSWORD con atributo tabindex.

Los cuadros de contraseña, como el resto de elementos de un formulario o una web en general programada en HTML, obtienen el foco según se van dando pulsaciones a la tecla tabulador según el orden en el que han sido dispuestas. Esto es muy útil para determinar, normalmente, el orden en el que se quieren incluir los elementos en la página, pero a veces es necesario incluir un orden visual y, a su vez, cambiar el orden de obtención de foco.

Para hacer esto tenemos el atributo TABINDEX, el cual define el orden de los elementos conforme se vayan accediendo a ellos para obtener el foco mediante la tabulación.



El resto de elementos del formulario que no tengan definidos este atributo conservarán el orden de tabulación normal.

## UD1

Veamos un ejemplo de un formulario con un cuadro de contraseña con este atributo:

```

<html>

    <head>
        <title>Ejemplo de cuadro de contraseña con el
        atributo tabindex.</title>
    </head>

    <body>

        <form action=>ingresar.php</action> method=>post</method>

        Usuario:
        <input type=>text</type> name=>nombree</name> tabindex=>2</tabindex> />
        <br/>

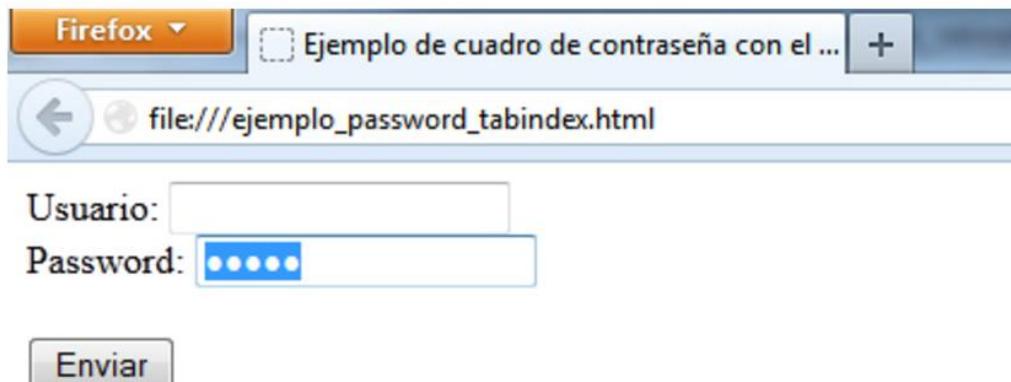
        Password:
        <input type=>password</type> name=>password</name> value=>12345</value>
        tabindex=>1</tabindex> />
        <br/>
        <br/>
        <input type=>submit</type> value=>Enviar</value> />
    </form>

</body>

</html>

```

Captura de pantalla del resultado que debe dar el código al ejecutarlo en un navegador web.



- Áreas de texto TEXTAREA con atributo rows.

En este punto entramos en los atributos de los campos de texto diseñados para contener textos de mucho mayor tamaño de los anteriores, como son las áreas de texto.

Dichos elementos tienen una barra de desplazamiento para moverse por ellos cómodamente conforme el usuario va introduciendo los datos.

A pesar de tener la barra de desplazamiento, dependiendo del uso que va a tener el área de texto a veces interesa que sea más grande o más pequeña.

En concreto, el número de líneas del área de texto por defecto es dos, pero se puede modificar mediante el uso del atributo ROWS, ya que el número que se le incluya en su valor será el número de líneas que tenga el área de texto.

Veamos un ejemplo de un formulario con un área de texto con este atributo:

```
<html>
<head>
    <title>Ejemplo de area de texto con el atributo rows.</title>
</head>
<body>
    <form action=>comentarios.php</action> method=>post</method>
        <area rows=>5</area>
    </form>
</body>
</html>
```

## UD1

Usuario:

```
<input type=>text</> name=>usuario</>
```

```
<br/>
```

Comentario:

```
<br/>
```

```
<textarea rows="5">Introduzca aqui sus comentarios...</>  
textarea>
```

```
<br/>
```

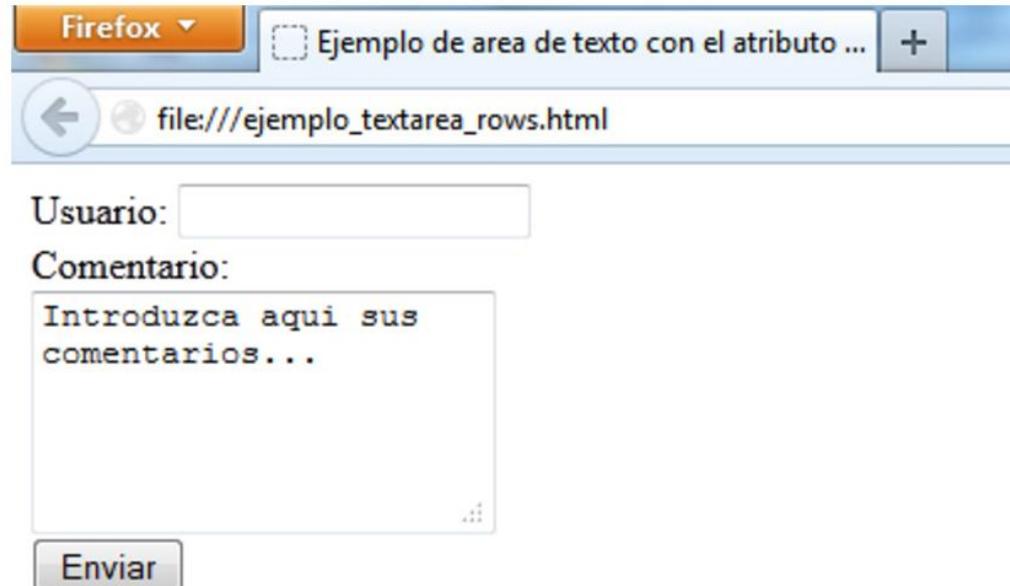
```
<input type="submit" value="Enviar"/>
```

```
</form>
```

```
</body>
```

```
</html>
```

Captura de pantalla del resultado que debe dar el código al ejecutarlo en un navegador web.



- Áreas de texto TEXTAREA con atributo cols.

Dentro de las áreas de texto, además de la importancia del número de líneas como hemos visto en el atributo anterior, la anchura que se le asigna a la misma también tiene relevancia.

Para introducir grandes cantidades de texto, que es la utilidad que tienen las TEXTAREA, a veces es importante que el cuadro donde se introduzca tenga el tamaño adecuado, no solo por su contenido, si no por la maquetación general de la página.

Para especificar dicha anchura en caracteres del área de texto, que al igual de los cuadros de texto por defecto es veinte, se usa el atributo COLS, donde el número que se le incluye en su valor será el número de caracteres que tiene de anchura el campo.

Veamos un ejemplo de un formulario con un área de texto con este atributo:

```
<html>
  <head>
    <title>Ejemplo de area de texto con el atributo cols.</title>
  </head>
  <body>
    <form action=>comentarios.php</action> method=>post</method>
      Usuario:
      <input type=>text</type> name=>usuario</name> />
      <br/>
      Comentario:
      <br/>
      <textarea cols=>50</cols> Introduzca aqui sus comentarios...</textarea>
      <br/>
```

## UD1

```

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

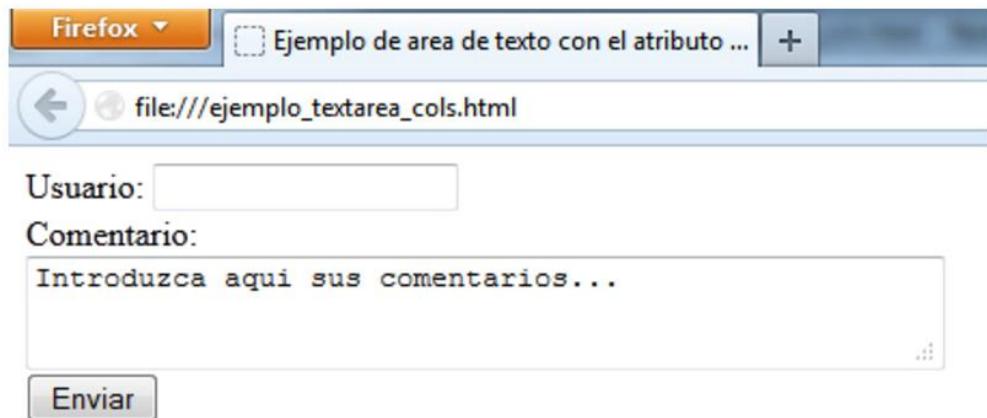
</form>

</body>

</html>

```

Captura de pantalla del resultado que debe dar el código al ejecutarlo en un navegador web.



- Áreas de texto TEXTAREA con atributo name.

Las áreas de texto, al igual que los cuadros de texto o de contraseña, también pueden ser referenciados por otros lenguajes de script como JavaScript. Para que ocurra esto, como ya hemos visto, se puede utilizar el atributo ID, pero también existe el atributo NAME, un atributo propio de HTML que cumple esta función y que además puede ser asignado a varios elementos.

Y sin olvidar que en el caso de los formularios, y en las áreas de texto en concreto, este atributo cobra mayor importancia si cabe, ya que sólo aquellos con un atributo name pasarán sus valores al servidor cuando se envíen los datos del formulario. Y no hay nada más frustrante para un usuario que introducir grandes cantidades de texto y que estas se pierdan.

Esto quiere decir que si un área de texto no tiene un atributo name asignado, aunque el usuario final lo rellene, no se enviará su información al servidor aunque se pulse un botón con la función de submit.

Veamos un ejemplo de un formulario con un área de texto con este atributo:

```
<html>

<head>

    <title>Ejemplo de area de texto con el atributo name.</title>

</head>

<body>

    <form action=>comentarios.php</action method=>post</method>

        Usuario:

        <input type=>text</type name=>usuario</name> />

        <br/>

        Comentario:

        <br/>

        <textarea name=>comentarios</name>>Introduzca aqui sus comentarios...</textarea>

        <br/>

        <input type=>submit</type value=>Enviar</value>/>

    </form>

</body>

</html>
```

Captura de pantalla del resultado que debe dar el código al ejecutarlo en un navegador web.

## UD1

The screenshot shows a Firefox browser window with the title bar "Ejemplo de area de texto con el atributo ...". The address bar displays "file:///ejemplo\_textarea\_name.html". The main content area contains a form. The first field is a text input with the label "Usuario:" and an empty input box. Below it is a text area with the label "Comentario:" and placeholder text "Introduzca aqui sus comentarios...". At the bottom of the form is a button labeled "Enviar".

- Áreas de texto TEXTAREA con atributo tabindex.

Las áreas de texto, como el resto de elementos de un formulario o una web en general programada en HTML, obtienen el foco según se van dando pulsaciones a la tecla tabulador según el orden en el que han sido dispuestas. Esto es muy útil para determinar, normalmente, el orden en el que se quieren incluir los elementos en la página, pero a veces es necesario incluir un orden visual y, a su vez, cambiar el orden de obtención de foco.

Para hacer esto tenemos el atributo TABINDEX, el cual define el orden de los elementos conforme se vayan accediendo a ellos para obtener el foco mediante la tabulación.



El resto de elementos del formulario que no tengan definidos este atributo conservarán el orden de tabulación normal.

Veamos un ejemplo de un formulario con un área de texto con este atributo:

```
<html>
<head>
```

```

<title>Ejemplo de area de texto con el atributo tabindex.</title>

</head>

<body>

<form action=>comentarios.php</action method=>post</method>>

    Usuario:

    <input type=>text</type name=>usuario</name tabindex=>2</tabindex> />

    <br/>

    Comentario:

    <br/>

    <textarea tabindex=>1</textarea>Introduzca aqui sus comentarios...</textarea>

    <br/>

    <input type=>submit</type value=>Enviar</value tabindex=>3</tabindex> />

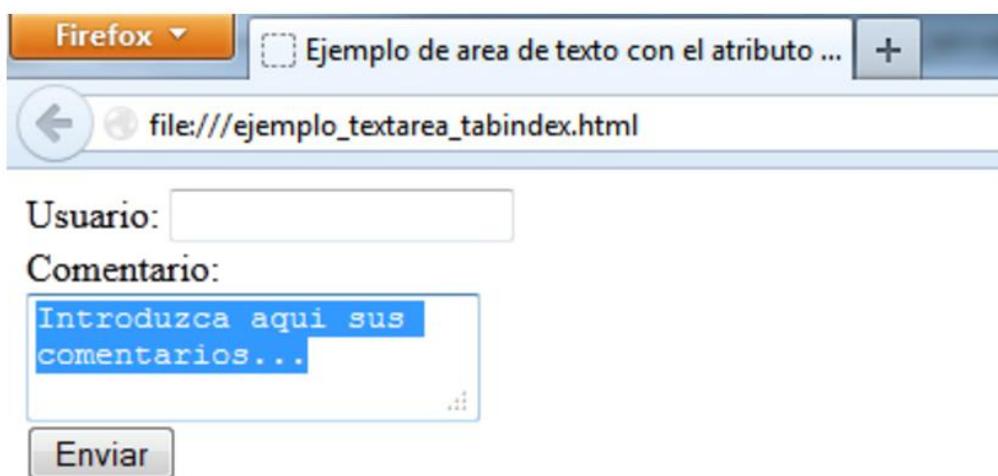
</form>

</body>

</html>

```

Captura de pantalla del resultado que debe dar el código al ejecutarlo en un navegador web.



## UD1

- Áreas de texto TEXTAREA con atributo wrap.

El último atributo en las áreas de texto que vamos a ver en esta Unidad Didáctica es el atributo WRAP. El cual define como va a interpretar las líneas y los saltos de línea el área de texto del formulario, tanto en su presentación como en su envío.

Hay tres posibles valores que puede tomar este atributo:

- “off”, que representa que mientras no se salte de línea todo esté escrito seguido.
- “virtual”, que representa el texto ajustado al ancho que tenga el área de texto, pero que cuando se envía al servidor lo hace en el mismo formato en el que se estén los saltos de línea, como si estuviese el atributo en off.
- “physical”, que ajusta el texto al ancho del área de texto y lo envía tal cual se ve.

Veamos un ejemplo de un formulario con un área de texto con este atributo:

```
<html>
<head>
    <title>Ejemplo de area de texto con el atributo wrap.</title>
</head>
<body>
    <form action=>>comentarios.php<> method=>>post<>>
        Usuario:
        <input type=>>text<> name=>>usuario<> />
        <br/>
        Comentario:
        <br/>
```

```

<textarea wrap="off">Introduzca aqui sus comentarios...</textarea>

<br/>

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

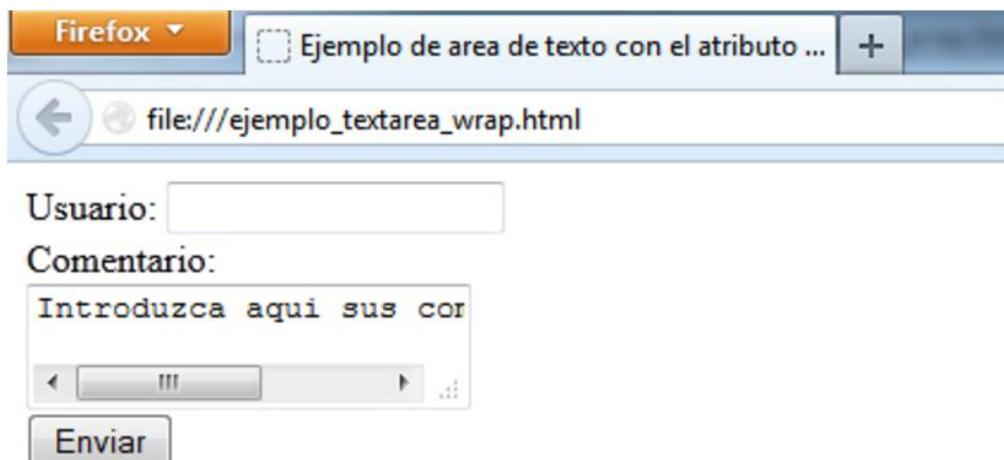
</form>

</body>

</html>

```

Captura de pantalla del resultado que debe dar el código al ejecutarlo en un navegador web.



En estos momentos el alumno debería ser capaz de diferenciar los tres campos de texto principales de un formulario:

- Los cuadros de texto.
- Los cuadros de contraseña.
- Y las áreas de texto.

Así como debería ser capaz de identificar, explicar y aplicar en el código todos los atributos principales de los mismos.

## UD1

A continuación vemos un ejemplo de un posible libro de visitas con los tres elementos, y todos los atributos relacionados que se han visto en esta Unidad Didáctica:

```

<html>
  <head>
    <title>Ejemplo de todos los campos de texto con sus
    atributos.</title>
  </head>
  <body>
    <form action=>libro_de_visitas.php</action> method=>post</method>
      Usuario:
      <input type=>text</type> size=>30</size> maxlength=>30</maxlength>
      name=>usuario</name> value=>Usuario</value> align=>left</align> tabindex=>1</tabindex>
    <br/>
      Password:
      <input type=>password</type> size=>10</size> maxlength=>10</maxlength>
      name=>password</name> value=>12345</value> align=>left</align> tabindex=>2</tabindex>
    <br/>
      Comentario en el libro de visitas:
    <br/>
      <textarea cols=>40</cols> rows=>5</rows> name=>comentarios</name> tabindex=>3</tabindex> wrap=>virtual</wrap>Introduzca aqui sus comentarios para el libro de visitas...</textarea>
    <br/>
      <input type=>submit</type> value=>Enviar</value>
    </form>
  </body>
</html>
```

Captura de pantalla del resultado que debe dar el código al ejecutarlo en un navegador web.

Firefox ▾ Ejemplo de todos los campos de texto c... +

file:///ejemplo\_c Campos\_texto.html

**Usuario:**

**Password:**

**Comentario en el libro de visitas:**

Introduzca aquí sus comentarios para el libro de visitas...

**Enviar**

#### 1.4. Formularios y eventos. Criterios de accesibilidad y usabilidad en el diseño de formularios

Los formularios, al igual que la mayoría de etiquetas HTML, pueden tener una interactividad aún más alta entre el sitio web en sí y el usuario final si se añaden eventos, que se ejecutan al cargar el código en el cliente.

Algunos de estos eventos ya los hemos visto, pero la lista completa es la siguiente:

- “onload”: Este evento es utilizado para activar alguna capacidad especial cuando el usuario termina de cargar una página.
- “onunload”: Este evento se activa cuando el usuario cierra o sale de formulario actual en su navegador web, ya sea en una ventana o una pestaña.
- “onclick”: Este evento se activa cuando se realiza click sobre el elemento que lo tiene.

## UD1

- “ondblclick”: Este evento es ejecutado cuando se hace doble click sobre el elemento.
- “onmousedown”: Este evento se activa cuando el puntero es presionado sobre el elemento.
- “onmouseup”: Este evento es activado cuando el puntero se suelta sobre el elemento.
- “onmouseover”: Este evento se activa cuando el puntero pasa sobre el elemento.
- “onmousemove”: Este evento se ejecuta cuando el puntero se mueve mientras está sobre el elemento.
- “onmouseout”: Este evento se activa cuando el puntero se quita de encima del elemento.
- “onfocus”: Este evento se ejecuta cuando el elemento recibe el foco, ya sea a través de un click de ratón, o a través del tabulador.
- “onblur”: Este evento se activa cuando el elemento pierde el foco, ya sea a través de un click de ratón, o a través del tabulador.
- “onkeypress”: Este evento se ejecuta cuando una tecla es presionada y luego soltada mientras el elemento tiene el foco.
- “onkeydown”: Este evento se ejecuta cuando una tecla es presionada mientras el elemento tiene el foco.
- “onkeyup”: Este evento se activa cuando se suelta una tecla mientras el elemento tiene el foco.
- “onsubmit”: Este evento se activa cuando el formulario es enviado.
- “onreset”: Este evento se ejecuta cuando el formulario restablece sus valores por defecto, cuando se pulsa un botón de reset.
- “onselect”: Ese evento se activa cuando un usuario selecciona texto en un campo de texto.
- “onchange”: Este evento se ejecuta cuando un elemento pierde el foco y su valor ha sido modificado desde que recibió el foco anteriormente.

Con estos eventos, y usando un lenguaje de script como JavaScript, el programador puede personalizar los formularios de forma visual con posibilidades casi infinitas.

Además de los eventos, que pueden darle una personalización especial al formulario, o incluso darle algunos usos de funcionalidad especial si se utilizan junto con lenguaje de scripts como por ejemplo JavaScript, hay otros criterios de presentación que se deben tener en cuenta.

La primera de estas herramientas es la agrupación de datos dentro de los formularios, que sirve para diferenciar los diversos grupos de datos que el usuario tiene que llenar dentro del mismo mediante un vistazo.

Por ejemplo, cuando se usa la búsqueda avanzada en el buscador de Google, al haber tantos datos en el formulario, están divididos en distintos grupos:

Firefox - Búsqueda avanzada de Google

https://www.google.es/advanced\_search?q=prueba&hl=es&sa=Search

Google

Búsqueda avanzada

Buscar páginas con...

todas estas palabras: prueba

esta palabra o frase exacta:

cualquier de estas palabras:

ninguna de estas palabras:

números desde el: hasta

Haz lo siguiente en el cuadro de búsqueda

Escribe las palabras importantes: terrier zotomero tricolor

Escribe las palabras exactas entre comillas: "terrier zotomero"

Escribe OR entre todas las palabras que quieras: ministra OR espiral

Añade un signo menos delante de las palabras que no quieras que aparezcan: -zodar, -"Jack Russell"

Escribe dos puntos seguidos entre los números y añade una unidad de medida: 10..35 kg, 300..600 euros, 2010..2011

A continuación, limitar los resultados por...

idioma: cualquier idioma

región: cualquier región

última actualización: en cualquier momento

sitio o dominio:

los términos que aparecen: en cualquier lugar de la página

SafeSearch: Mostrar los resultados más relevantes

Captura de pantalla de la página de búsqueda avanzada del buscador de Google en el navegador Mozilla Firefox.

Podemos ver como el programador ha optado por dividir el formulario en dos grupos, el primero le pide al usuario que rellene los cuadros de texto con información relacionada con las palabras que quiere encontrar en la búsqueda, y el segundo tiene diversas listas desplegables para que el usuario pueda limitar las búsquedas en diversos factores fijos.

## UD1

En realidad todos los elementos pertenecen al mismo formulario, pero el hecho de que estén separados en grupos facilita la visualización y uso del formulario al usuario final.

Otro detalle que sirve para facilitar la visualización al usuario es la división en distintas páginas del mismo. Esto se hace, simplemente, haciendo varios formularios en distintas páginas web, pero cuando se pulsa el botón de submit, si todo es correcto, el formulario que se presentará en pantalla es el siguiente.

También es importante que se identifique claramente, de cara al usuario, que campos tiene que llenar obligatoriamente y cuales no. Esto se puede hacer de diversas formas. Por ejemplo, cuando se quiere crear un nuevo correo en la página web de Yahoo, indican que se puede introducir un número de teléfono de recuperación opcional:

Captura de pantalla de la página de creación de cuenta de correo en la página web de Yahoo España en el navegador Mozilla Firefox.

Pero generalmente esto se informa al usuario por otras vías, como por ejemplo poner un asterisco junto a los campos obligatorios, e indicarlo en la parte inferior de la página.

En todo caso, el programador debe tener claro que los datos deben mostrarse ante el usuario final siempre lo más ordenados posibles. Por ejemplo, no tiene sentido pedir a un usuario su nombre propio, su fecha de nacimiento, su di-

rección y a continuación pedirle sus apellidos, ya que una ordenación de esa forma sería ilógica y agreste para cualquier usuario.

Además, los datos que se piden deben mostrarse claros, y sin ningún tipo de dudas de lo que se está pidiendo al mismo. En el caso de que haya dos formas de expresar o de pedir un dato en el formulario, y una de las dos sea más clara y genere menos dudas siempre se debe optar por esa opción.

En definitiva, el programador debe ser consciente, además de las soluciones técnicas que pueda aportar, de que el usuario final no va a tener ningún conocimiento informático.

#### 1.4.1. Agrupación de datos

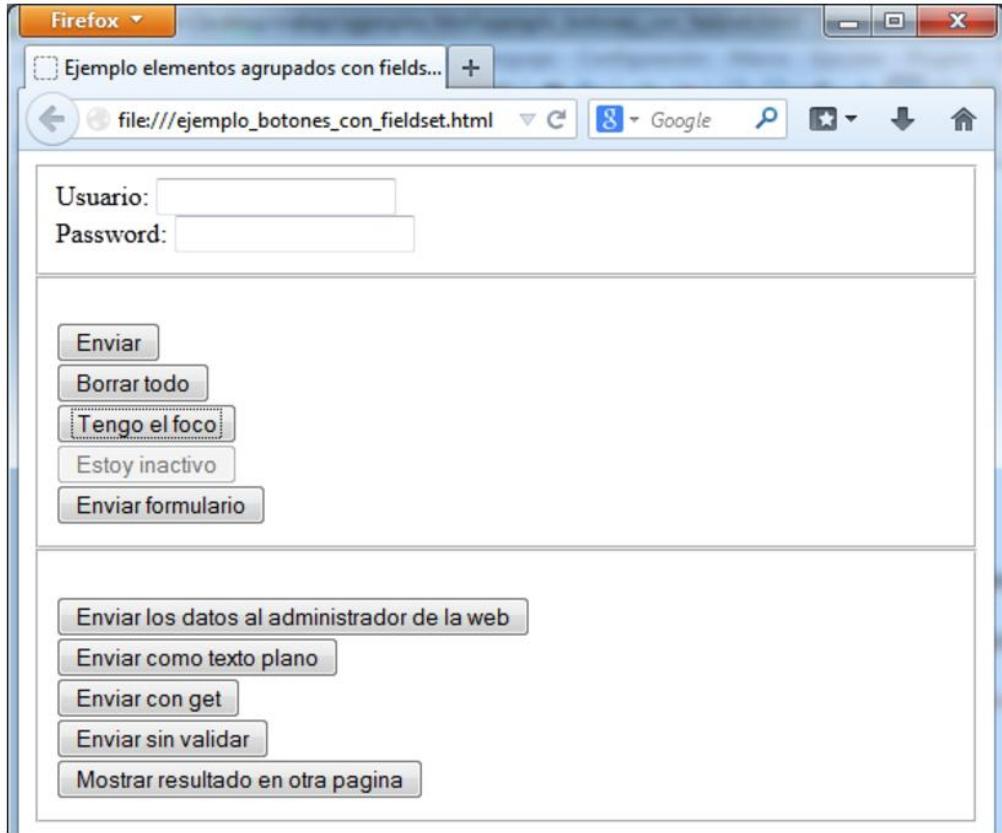
Como ya hemos visto en el punto anterior, agrupar los datos en un formulario es un buen sistema para facilitar el ingreso de los datos al usuario final, y para que este pueda identificar las distintas partes del formulario de un simple vistazo.

Para hacer esto, en HTML se usa una etiqueta en concreto, que es la FIELDSET. Como la mayoría de etiquetas HTML se trata de una etiqueta que requiere apertura y cierre, y todo lo que está contenido a su alrededor se considera agrupado.

Los elementos agrupados mediante el uso de esta herramienta son agrupados en distintas cajas, para diferenciarlos a unos de otros. Por ejemplo, si tenemos muchos tipos de botones y los agrupamos mediante la etiqueta FIELDSET el resultado tendría este aspecto:

Captura de pantalla de un ejemplo de un formulario con muchos botones agrupados mediante el elemento FIELDSET.

## UD1



Los atributos más comunes e importantes del elemento FIELDSET son los siguientes:

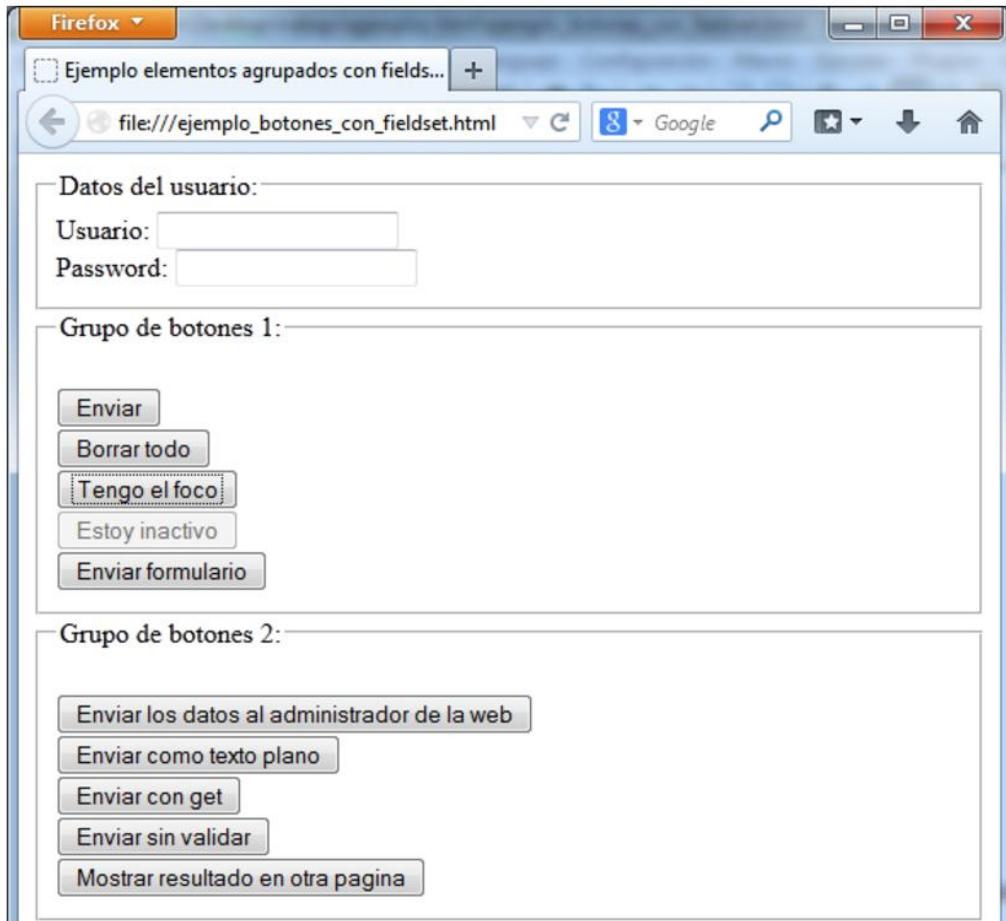
- Disabled.
- Form.
- Name.

Los cuales veremos en profundidad a continuación, junto con ejemplos en código HTML para cada uno de ellos.

Por supuesto, el uso de esta herramienta para separar los datos dentro de un mismo formulario tiene muchísima utilidad, pero tiene un inconveniente, y es que no permite nombrar a cada grupo de forma que se les pueda identificar. Sin embargo, existe un elemento que permite hacer eso, ni más ni menos. Se trata de la etiqueta LEGEND.

La etiqueta LEGEND necesita, al igual que FIELDSET, una etiqueta de apertura y otra de cierre. Esta etiqueta no tiene sentido sin que esté incluida en la anterior. Por lo que, simplemente se usa para nombrar cada una de las agrupaciones de forma que se facilite, aún más, la identificación de cada uno de esos grupos de cara al usuario.

Por ejemplo, aquí tenemos la imagen anterior, pero ahora usando también la etiqueta LEGEND en cada una de las agrupaciones de FIELDSET.



Captura de pantalla de un ejemplo de un formulario con muchos botones agrupados mediante el elemento FIELDSET y nombrados usando el elemento LEGEND.

El único atributo relevante de la etiqueta LEGEND es el ALIGN, el cual veremos en profundidad a continuación junto con un ejemplo de código.

## UD1

- Agrupación de elementos de un formulario con FIELDSET con el atributo disabled.

El atributo DISABLED es bastante especial.

Lo primero que se puede decir de él es que es un atributo booleano. Por lo tanto solo tiene dos posibles valores, estar activado o no. Para ello debe estar presente en la declaración del elemento FIELDSET o no.

Lo segundo es que una agrupación con este atributo activo está deshabilitado, y ninguno de los elementos en su interior puede ser usado hasta que se activa. Por lo tanto es un atributo con el que se debe jugar con otros lenguajes como con JavaScript o PHP.

Veamos un ejemplo de una agrupación de elementos de formulario con este atributo:

```
<html>
<head>
    <title>Ejemplo de etiqueta FIELDSET con el atributo
    disabled.</title>
</head>
<body>
    <form action=>formulario.php</action> method=>post</method>
    <fieldset disabled>
        <legend>
            Datos personales
        </legend>
        Nombre:
        <input type=>text</type> name=>nombre</name> value=></value>
        <br/>
        Apellidos:
        <input type=>text</type> name=>apellidos</name> value=></value>
```

```

<br/>

<input type=>submit name=>enviar value=>Enviar>
/>

</fieldset>

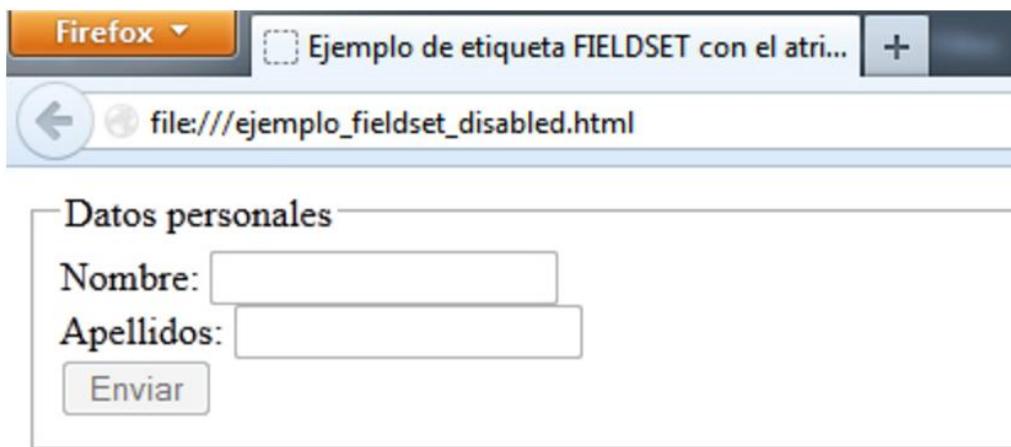
</form>

</body>

</html>

```

Captura de pantalla del resultado que debe dar el código al ejecutarlo en un navegador web.



- Agrupación de elementos de un formulario con FIELDSET con el atributo form.

El atributo form en una agrupación de elementos de un formulario tiene una utilidad muy importante, ya que permite que, a pesar de que la agrupación se declare en un lugar del código fuera del formulario, aún así pueda seguir unida a este.

Esto hace que podamos declarar un formulario en una parte distinta del código de una página web, programar otros elementos, y donde quiera el programador poner una agrupación de elementos que serán enviados junto con el formulario cuando se le dé al botón de submit.

Veamos un ejemplo de una agrupación de elementos de formulario con este atributo:

## UD1

```

<html>
  <head>
    <title>Ejemplo de etiqueta FIELDSET con el atributo form.</title>
  </head>
  <body>
    <form      action=>>formulario.php<>      method=>>post<>
      id=>>formulario<>>
        Usuario:
        <input type=>>text<> name=>>usuario<> value=>>> />
        <br/>
        <input type=>>submit<> name=>>enviar<> value=>>Enviar<>
      />
    </form>
    <fieldset form=>>formulario<>>
      <legend>
        Datos personales
      </legend>
      Nombre:
      <input type=>>text<> name=>>nombre<> value=>>> />
      <br/>
      Apellidos:
      <input type=>>text<> name=>>apellidos<> value=>>> />
    </fieldset>
  </body>
</html>

```

Captura de pantalla del resultado que debe dar el código al ejecutarlo en un navegador web.

**Firefox** ▾ Ejemplo de etiqueta FIELDSET con el atributo name +

file:///ejemplo\_fieldset\_form.html

Usuario:

**Enviar**

**Datos personales**

Nombre:

Apellidos:

- Agrupación de elementos de un formulario con FIELDSET con el atributo name.

Al igual que ocurre normalmente con el resto de elementos en lenguaje HTML, el atributo NAME sirve para asignarles a las agrupaciones de elementos un nombre para poder ser referenciado por otros lenguajes como JavaScript.

Realmente la funcionalidad de los atributos NAME e ID son parecidas, pero la ventaja de usar el atributo NAME es que se le puede dar a varias agrupaciones de elementos el mismo nombre, y cuando se programa en otro lenguaje y se hace referencia al elemento con ese nombre, todos los elementos de las agrupaciones que tengan el mismo se verán afectadas.

Veamos un ejemplo de una agrupación de elementos de formulario con este atributo:

```
<html>
<head>
    <title>Ejemplo de etiqueta FIELDSET con el atributo name.</title>
</head>
<body>
    <form action=>formulario.php</action> method=>post</method>
        <FIELDSET name=>datosPersonales</FIELDSET>
            <label>Nombre:</label> <input type=>text</type>
            <label>Apellidos:</label> <input type=>text</type>
        </FIELDSET>
        <input type=>submit value=>Enviar</type>
    </form>
</body>
```

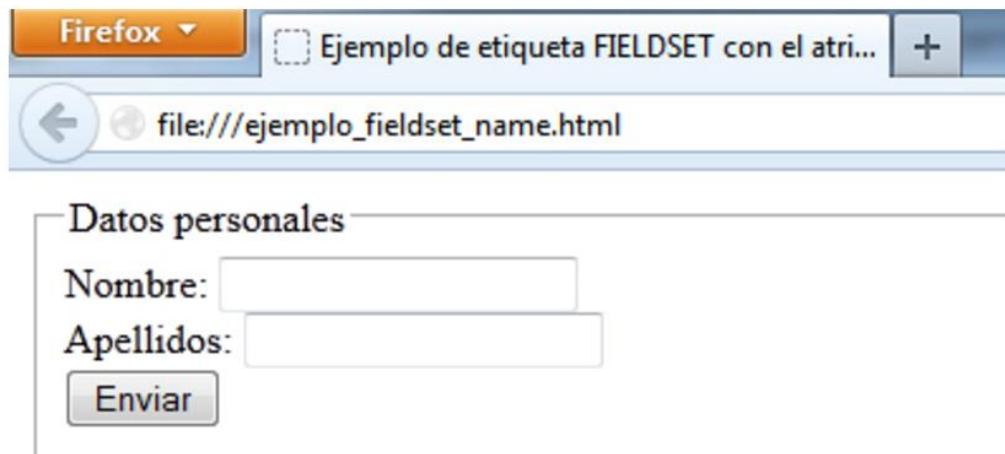
## UD1

```

<fieldset name=></a>
    <legend>
        Datos personales
    </legend>
    Nombre:
    <input type=><a href="#">text</a></input> name=><a href="#">nombre</a> value=></a>
    <br/>
    Apellidos:
    <input type=><a href="#">text</a></input> name=><a href="#">apellidos</a> value=></a>
    <br/>
    <input type=><a href="#">submit</a></input> name=><a href="#">enviar</a> value=><a href="#">Enviar</a>
/>
</fieldset>
</form>
</body>
</html>

```

Captura de pantalla del resultado que debe dar el código al ejecutarlo en un navegador web.



- Nombrar agrupaciones de elementos de formulario con LEGEND con el atributo align.

Cuando se usa el elemento LEGEND se pone un texto en la parte superior de la agrupación de elementos de FIELDSET. Pero a veces, el lugar donde está colocado este nombre puede resultar, de cara al diseño, interesante que esté en otro lugar.

Para ello nos encontramos con el atributo ALIGN, que define dónde está ese nombre de la agrupación que hemos definido con este elemento.

El alineamiento puede ser:

- Top, para que se muestre en la parte superior.
- Bottom, para que se muestre en la forma inferior.
- Left, para que se muestre a la izquierda, que es la que tiene por defecto si no añadimos este atributo.
- Right, para que se muestre a la derecha.

Veamos un ejemplo de una agrupación de elementos de formulario con este atributo:

```
<html>
<head>
    <title>Ejemplo de etiqueta LEGEND con el atributo align.</title>
</head>
<body>
    <form action=>formulario.php</action> method=>post</method>
        <fieldset name=>agrupacion1</name>
            <legend align=>right</align>
                Datos personales
            </legend>
            Nombre:
</body>
```

## UD1

```

<input type=>text</> name=>nombre</> value=></>
<br/>

Apellidos:

<input type=>text</> name=>apellidos</> value=></>
<br/>

<input type=>submit</> name=>enviar</> value=>Enviar</>
/>

</fieldset>

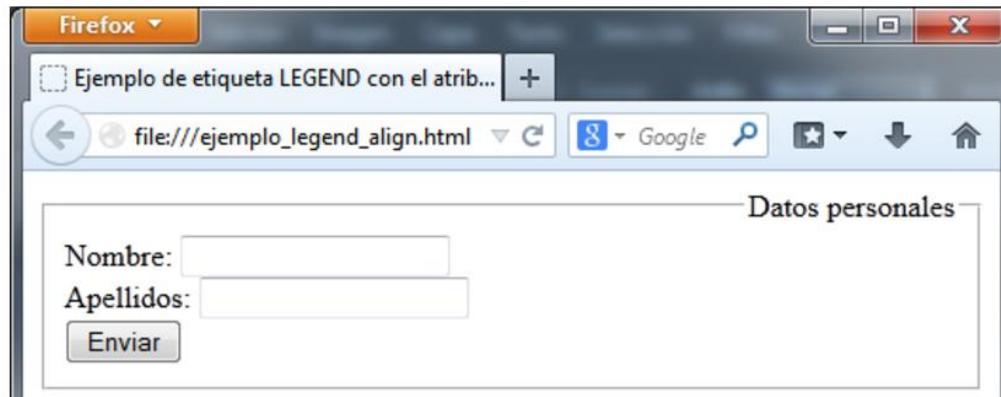
</form>

</body>

</html>

```

Captura de pantalla del resultado que debe dar el código al ejecutarlo en un navegador web.



Y finalmente, con estos cuatro atributos de los dos elementos para hacer agrupaciones de elementos, FIELDSET y la etiqueta para nombrarlas LEGEND, el programador puede hacer separaciones lógicas y funcionales de los datos con la intención de que el usuario final tenga menos dificultades para llenar el formulario.

A continuación tenemos un ejemplo con todos estos atributos:

```

<html>
<head>
    <title>Ejemplo de elementos FIELDSET y LEGEND con todos los atributos.</title>
</head>
<body>
    <form      action=>>formulario.php<>      method=>>post<>
        id=>>formulario<>>
        <fieldset name="datos_personales">
            Nombre:
            <input type="text" name="nombre" value="" />
            <br/>Apellidos:
            <input type="text" name="apellidos" value="" /><br/>
        </fieldset>
        <fieldset disabled>
            Usuario:
            <input type="text" name="usuario" value="" /><br/>
            <input type="submit" name="enviar" value="Enviar" />
        </fieldset>
    </form>
    <br/>
    <fieldset form="formulario">
        <legend align="right">
            Comentarios:

```

## UD1

```

</legend>

<textarea></textarea>

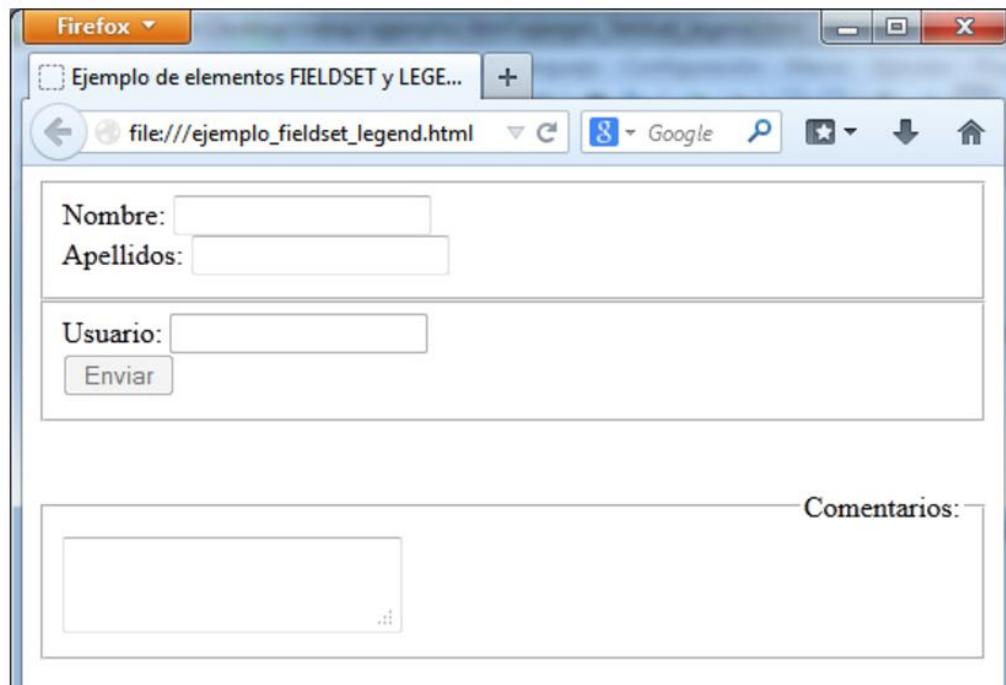
</fieldset>

</body>

</html>

```

Captura de pantalla del resultado que debe dar el código al ejecutarlo en un navegador web.



#### 1.4.2. Adecuación del tamaño del formulario (división en distintas páginas)

Normalmente la mayoría de formularios web son de poca longitud y con muy pocos elementos. Están hechos para ser totalmente funcionales sin abrumar al usuario final con decenas de campos para llenar, y la simpleza unida a la efectividad es un valor añadido para cualquier página web en la actualidad.

Sin embargo, a veces los formularios necesitan tener varias páginas, ya sea porque son muchísimos los datos que es necesario introducir, o simplemente

porque es necesario un paso previo antes de acceder a un formulario, y en ese dato previo hay otro formulario que debe ser llenado correctamente para acceder al siguiente.

Por ello, la división en distintas páginas del formulario es esencial.

De todas maneras, el programador debe ser consciente de las peculiaridades de su diseño antes de hacerlo. Y por supuesto tener claro que sólo se debe dividir este en distintas páginas si se dan las siguientes situaciones:

- El formulario es demasiado largo y tiene muchos campos.
- Es necesario el paso por un formulario previo antes de acceder al formulario o formularios finales.

Por ejemplo, en la página de facebook hay que loguearse con el siguiente formulario:



Captura de pantalla del formulario de ingreso en la página principal de Facebook con el navegador Mozilla Firefox.

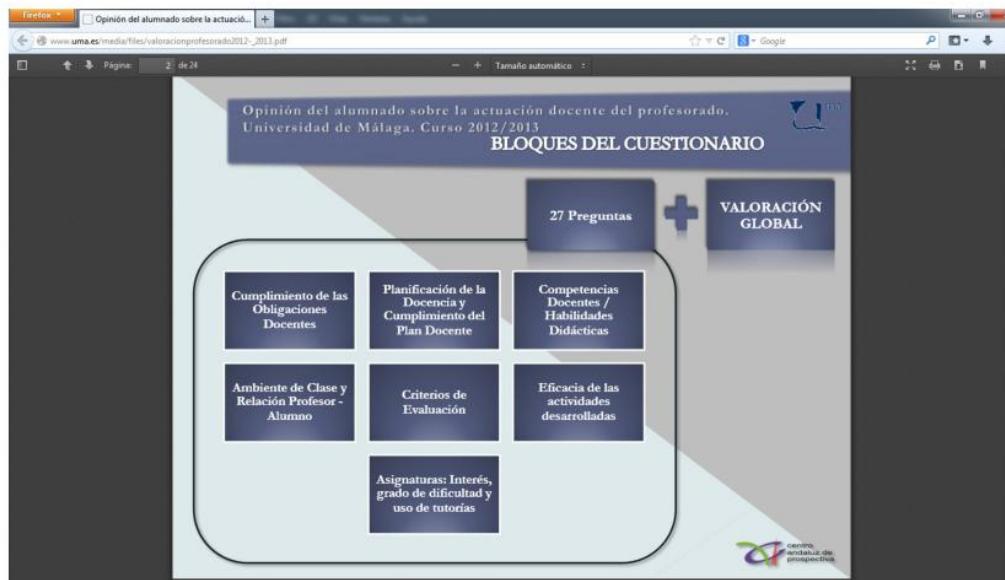
Y una vez que se ha introducido correctamente el usuario y la contraseña, se podrá usar este otro formulario de búsqueda:



Captura de pantalla del formulario de búsqueda en la página de entrada de Facebook con el navegador Mozilla Firefox.

Y por ejemplo, una encuesta sobre la valoración del profesorado, como la que realiza todos los años la Universidad de Málaga, y que podemos ver el resultado en su página web:

## UD1



Captura de pantalla de la metodología de la encuesta del profesorado del curso 2012/2013 que se realiza anualmente en la Universidad de Málaga con el navegador Mozilla Firefox.

Que cuenta con 27 preguntas, más una valoración global final, es muy poco recomendable hacerla vía web en una sola página, ya que poder navegar por ella sería extremadamente farragoso, e incluso consultar respuestas anterior.

Por todo esto, el programador debe ser muy consciente de cuando usar una solución como esta cuando sea necesario.

En todo caso, ambos formularios deben ser programados en páginas separadas.

Veamos un ejemplo:

```
<html>
<head>
<title>Ejemplo de formulario de ingreso.</title>
</head>
<body>
<form action=>ingreso.php</action> method=>post</method>
```

```

    Usuario: <input type=>text</> size=>30</> maxlength=>30</>
    name=>usuario</> value=>Usuario</> /> <br/>

    Password: <input type=>password</> size=>10</> maxleng-
    th=>10</> name=>password</> /> <br/>

    <input type=>submit</> value=>Entrar</> />

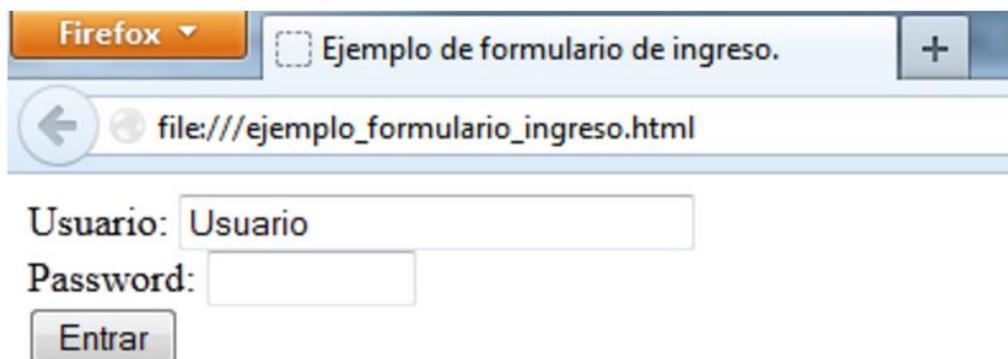
</form>

</body>

</html>

```

Captura de pantalla del resultado que debe dar el código al ejecutarlo en un navegador web.



Si el servidor comprueba que los datos de registro son correctos, el usuario podrá acceder al buscador en la segunda página:

```

<html>
  <head>
    <title>Ejemplo de buscador.</title>
  </head>
  <body>
    <form action="buscador.php" method="post">
      Introducir datos a buscar: <br/>
      <input type="text" name="buscador" /> <br/><br/>
    </form>

```

## UD1

```

<input type="submit" value="Buscar"/>

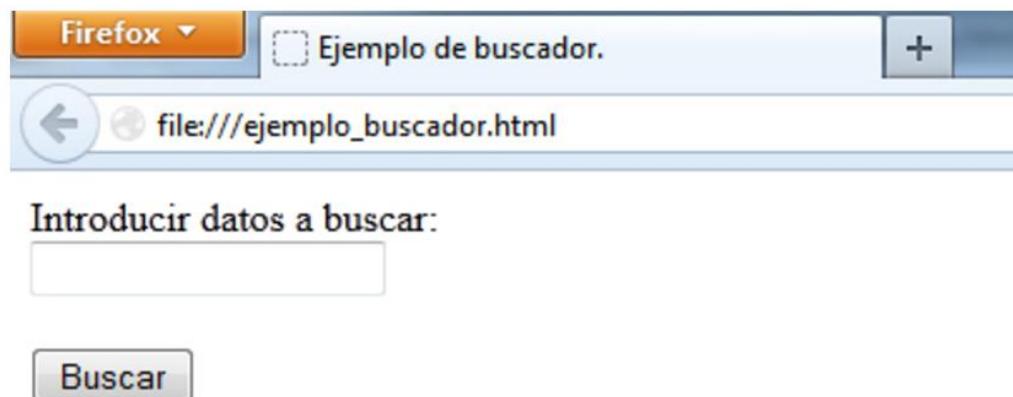
</form>

</body>

</html>

```

Captura de pantalla del resultado que debe dar el código al ejecutarlo en un navegador web.



### 1.4.3. Identificación de los campos obligatorios

En los formularios, tanto en los físicos como en los formularios web, normalmente cuando se incluye un campo en el mismo es porque se quiere que el usuario final lo complete. No tiene sentido, en principio, añadir un campo que no es necesario para la recopilación de datos del formulario.

Sin embargo, a veces hay datos en un formulario que son realmente vitales para la información que se quiere reunir, y otros que, aunque son interesantes, no lo son tanto.

Para cuando ocurren estos casos, el programador puede diferenciar entre incluir algunos campos como obligatorios y otros que no lo sean.

Por ejemplo, en un formulario de registro, como mínimo lo normal es que el nombre de usuario, la contraseña y el correo electrónico del mismo sean datos obligatorios, pero no así su edad, ciudad natal o fecha de nacimiento.

Cómo identificar estos campos obligatorios es otra de las cuestiones. Normalmente se suele hacer mediante el uso de asteriscos junto a los campos que son de obligada inclusión. Pero hoy en día se adoptan otras soluciones igual de imaginativas. En todo caso, el usuario final debe poder comprobar, de un simple vistazo, cuales son los campos obligatorios y cuales no. Por ejemplo, en la web de la Liga de Fútbol Profesional Española, el formulario de registro identifica los campos obligatorios con asteriscos:

Captura de pantalla de la página web de registro de la Liga de Fútbol Profesional Española en el navegador Mozilla Firefox. Como se puede comprobar, el nombre y el primer apellido es un campo obligatorio, pero el segundo apellido no lo es.



Aunque la Liga de Fútbol Profesional se creó en 1984 como tal, la liga española empezó a disputarse en 1929.

Pero, además de indicar al usuario cuales son los campos obligatorios, ya sea mediante el uso de texto plano o del atributo VALUE de los campos de texto, hay varias formas de programar para que el formulario no se pueda enviar si no se rellenan los campos obligatorios:

## UD1

- Programando, de cara al servidor, que sean obligatorios que estén rellenos ciertos campos para que el formulario sea aceptado.
- Programando en el cliente añadiendo a las etiquetas de campos obligatorios un atributo para indicar que son obligatorias.

Este atributo es el REQUIRED, el cual, si se le añade a un elemento de un formulario, este debe ser llenado, obligatoriamente, o al pulsar cualquiera de las opciones de SUBMIT del formulario, saldrá un mensaje junto con dicho campo indicando que debe ser completado.

Por ejemplo, en un formulario de ingreso, es obvio que tanto como el usuario y la contraseña deberían tener el atributo REQUIRED.

Sin embargo, en un formulario de registro, quizás no es necesario que el campo de texto de la dirección tenga el atributo REQUIRED.

En todo caso, la decisión depende del programador.

El programador, siempre, debe ser consciente en sus diseños de la finalidad funcional de los mismos. Ya que no es lo mismo estar diseñando un formulario de ingreso para una página web casual, como puede ser la de un foro anónimo en internet, que el formulario de ingreso en una web oficial como la de la agencia tributaria.

Obviamente en el segundo caso, los datos a llenar deberían ser muchos más, y los obligatorios también. En el caso del foro anónimo, mientras menos datos se pidan al usuario, mucho mejor.

Por ejemplo, veamos el código de un formulario de un libro de visitas donde sólo es obligatorio llenar el área de texto, ya que el usuario y la contraseña a llenar son opcionales. Esto quiere decir que, si no se llenan, el nombre del usuario que ha introducido el comentario aparecerá como anónimo. Pero claro, el comentario siempre será obligatorio llenarlo, sea un usuario anónimo o no:

```
<html>
<head>
    <title>Ejemplo de campos obligatorios.</title>
</head>
<body>
```

```

<form action=>libro_de_visitas.php</action method=>post</method>>

  Usuario:

    <input type=>text</type> size=>30</size> maxlength=>30</maxlength>
    name=>usuario</name> value=>Usuario</value> align=>left</align> tabindex=>1</tabindex>
  />

  <br/>

  Password:

    <input type=>password</type> size=>10</size> maxlength=>10</maxlength>
    name=>password</name> value=>12345</value> align=>left</align> tabindex=>2</tabindex>
  />

  <br/>

  Comentario en el libro de visitas:

  <br/>

  <textarea cols=>40</cols> rows=>5</rows> name=>comentarios</name> tabindex=>3</tabindex> required>Introduzca aqui sus comentarios para el libro de visitas...</textarea>

  <br/>

  <input type=>submit</type> value=>Enviar</value> />

</form>

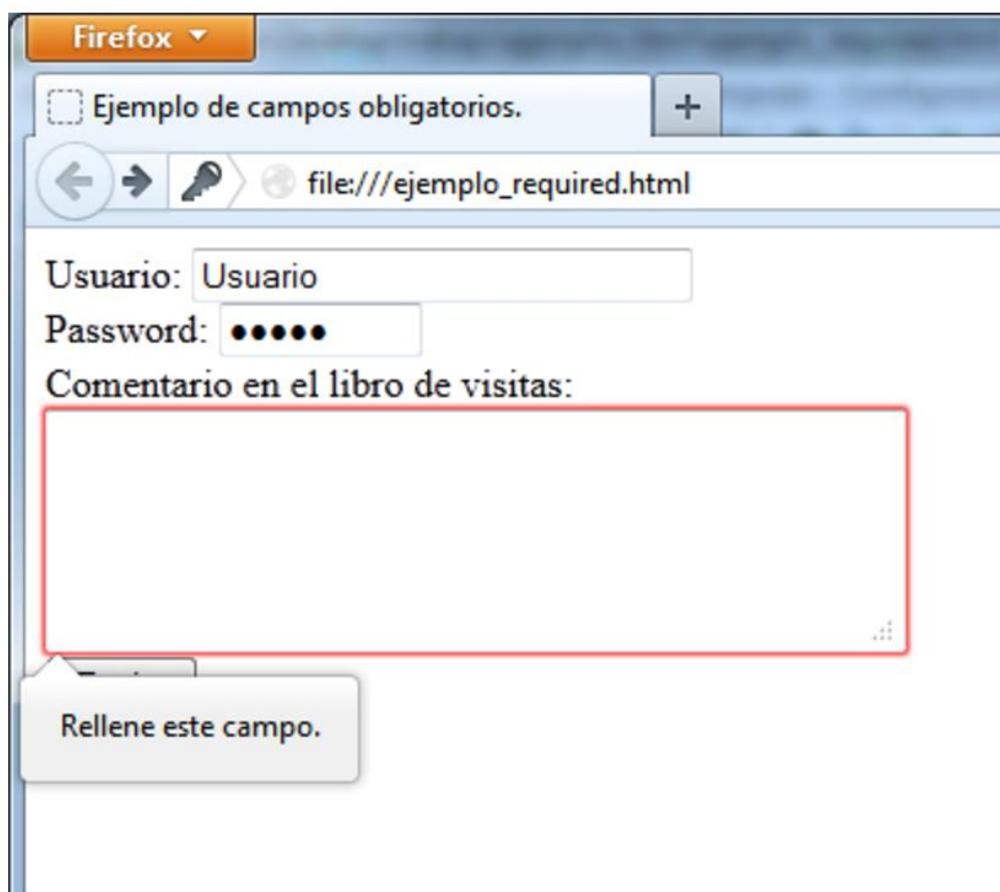
</body>

</html>

```

Captura de pantalla del resultado que debe dar el código al ejecutarlo en un navegador web.

## UD1



Véase que al intentar mandar el formulario con el campo con el atributo REQUIRED sin nada escrito, da un error.

#### 1.4.4. Ordenación lógica de la petición de datos

Una de las máximas de cualquier elemento web que se hace hoy en día es que sea lo más agradable a la vista posible para el usuario final. Esto se consigue mediante diversas técnicas:

- Procurar que los elementos que van uno a continuación de otros en la vida real lo hagan así en los formularios. Por ejemplo, los apellidos después del nombre.
- Los datos más relevantes deben estar destacados, y a ser posible arriba del todo.

- Los campos con tamaños adecuados para ponerse uno al lado del otro, porque son más pequeños de lo habitual, deben colocarse juntos, siempre que no entren en conflicto con los dos puntos anteriores.
- Las listas desplegables o cualquier elemento que se despliegue no debe tapar otros elementos que pueden ser esenciales a la hora de elegir el valor que tiene que escoger el usuario. En concreto si la lista desplegable tapa el texto que identifica que es lo que debe escoger el usuario, puede darse la situación, poco elegante a nivel funcional, de que el usuario tenga que salir de la lista para poder ver de nuevo que es lo que tiene que elegir.
- Los elementos relacionados tienen que estar lo más agrupados entre sí posible. Un formulario muy mal diseñado tendría, por ejemplo, los datos personales por un lado, a continuación otro tipo de datos, y acto seguido la petición al usuario de otro dato personal, por ejemplo la fecha de nacimiento. Esto confunde al usuario, y sobre todo, deja una muy mala impresión del formulario.
- Los botones deben estar siempre en el lugar más adecuado, que suele ser, salvo excepciones, al final del documento. No tiene sentido que el botón de enviar la información del formulario, por ejemplo, se encuentre en la mitad del mismo, ya que si el usuario le diese aún habría gran parte de la información necesaria, en este caso la mitad, que el usuario no habría completado. Además, es muy recomendable que el botón de enviar se encuentre a la derecha del botón de borrar la información, y que ambos se diferencien muy claramente. No hay nada más frustrante para un usuario que llenar toda la información de un formulario muy extenso y luego, por error, darle al botón de borrar todos los datos del formulario.



La ordenación de los formularios web donde se introducen los datos personales del usuario no difiere mucho de los formularios físicos, por lo que el programador puede recurrir a uno de ellos para ordenar los datos en un formulario similar. Sin embargo hay que tener en cuenta que muchos elementos que existen en los formularios web no existen en estos formularios, como son las listas desplegables y los botones de opción, por lo que siempre se debe elegir la opción más correcta cuando sea posible usar este tipo de elementos. Y sobre todo, no hay que olvidar nunca que en un formulario web siempre hay que añadir al final al menos un botón para enviar la información, algo que, obviamente, no necesita un formulario físico.

## UD1

Por ejemplo, si se quiere hacer una reclamación anónima al Ministerio del Interior del Gobierno de España a través de internet se accede al siguiente formulario:

La captura de pantalla muestra un formulario web titulado "Solicitud de información". El formulario es para la "RECLAMACIÓN ANÓNIMA". Los campos obligatorios marcados con (\*) son: Asunto (\*), Email (\*), Texto (\*). Hay un cuadro grande para escribir el texto de la reclamación. Se indica que se permite adjuntar un archivo. Abajo, se pide introducir los caracteres visibles en la imagen, que son "som eday". A la derecha, hay botones para "Borrar Formulario" y "Enviar".

Captura de pantalla del formulario de reclamaciones anónimas vía web del Ministerio del Interior del Gobierno de España en el navegador Mozilla Firefox.

Como se puede observar, lo primero que hay que rellenar es el asunto y el email, y a continuación el cuerpo del mensaje. Finalmente, a la derecha en la parte inferior están los botones para borrar los datos del formulario y para enviarlo.

Por lo tanto, cuando el programador diseñe un formulario web, debe tener en cuenta todas estas variables para hacer un formulario lo más adecuado posible.

En todo caso, los formularios más típicos, y que además requieren una ordenación de datos lo más correcta posible son los formularios que solicitan los datos personales.

Veamos un ejemplo de cómo se programaría uno de ellos:

```
<html>
<head>
```

## UF1304: Elaboración de plantillas y formularios

```

<title>Ejemplo de formulario que requiere datos personales.</title>

</head>

<body>

<form action="datos_personales.php" method="post">

<table>

<tr>

<td>Nombre: </td><td><input type="text" name="nombre" required /> </td>

</tr><tr>

<td>Apellidos: </td><td><input type="text" name="apellidos" required /> </td>

</tr><tr>

<td>DNI: </td><td><input type="text" name="dni" required /> </td>

</tr><tr>

<td>Domicilio: </td><td><input type="text" name="domicilio" /> </td>

</tr><tr>

<td>Poblacion: </td><td><input type="text" name="poblacion" /> </td>

</tr><tr>

<td>Codigo Postal: </td><td><input type="text" name="cp" /> </td>

</tr><tr>

<td>Email: </td><td><input type="text" name="email" /> </td>

</tr><tr>

<td>Telefono: </td><td><input type="text" name="telefono" /> </td>

```

## UD1

```
</tr><tr>  
<td></td><td><input type="submit" value="Enviar" /></td>  
</tr>  
</table>  
</form>  
</body>  
</html>
```

The screenshot shows a Firefox browser window with the title bar "Ejemplo de formulario que requiere dato..." and the address bar "file:///ejemplo\_formulario\_datos\_personales.html". The main content area displays a form with the following fields:

Nombre:	<input type="text"/>
Apellidos:	<input type="text"/>
DNI:	<input type="text"/>
Domicilio:	<input type="text"/>
Poblacion:	<input type="text"/>
Código Postal:	<input type="text"/>
Email:	<input type="text"/>
Teléfono:	<input type="text"/>

Below the input fields is a blue "Enviar" button.

Captura de pantalla del resultado que debe dar el código al ejecutarlo en un navegador web.

#### 1.4.5. Información correcta al usuario.

Cuando un usuario rellena los campos de un formulario simplemente está utilizando una poderosa herramienta de interactividad con el código que haya creado el programador.

Pero, si el usuario introduce información incorrecta, toda la utilidad del formulario desaparece completamente.

Por lo tanto, además de ordenar los datos de forma correcta para que el usuario no se confunda, es necesario cumplir una máxima siempre que se hace cualquier tipo de programa, que no es ni más ni menos que asumir que el usuario final no tiene el más mínimo conocimiento de informática y no sabe que es lo que tiene que llenar en cada campo.



Importante

Siempre que se hace un programa debe hacerse pensando que el usuario final va a ser el posible usuario, entre todos los que van a acceder a la aplicación, más torpe posible. Es decir, si tenemos que hace un formulario y es posible que lo tenga que acabar llenando una persona de alta edad con escasos conocimientos de informática, hay que pensar en como va a ver esa persona el formulario y cada uno de los campos, y adecuarlos para ese tipo de persona, ya que si el usuario más torpe de una aplicación web es capaz de utilizarla, cualquier otro usuario es seguro que también podrá hacerlo.

Por lo tanto, a la hora de identificar un campo se debe dejar muy claro que es lo que el programador pretende que el usuario final rellene en ese lugar.

Por ejemplo, si diseñamos un formulario de registro en una página web de compras online, donde normalmente se van a requerir datos personales, datos de envío de productos e incluso métodos de pago, es importante que cada uno de los campos de texto estén debidamente asociados a los textos relacionados con su contenido, y que además dejen claro sin ningún género de dudas qué es lo que hay que poner en cada campo.

El ejemplo clásico es el del botón de "reset" de los formularios. Podemos encontrarnos con múltiples formas de llamar a estos botones en la red, usando el atributo value. Sin embargo algunos de ellos tendrían serios problemas si el formulario lo encontrase una persona con nulos conocimientos de informática.

## UD1

Ya que no es lo mismo un posible valor de "borrar todo el formulario", donde queda meridianamente claro qué es lo que hace el botón, u otro posible valor de "restablecer", donde un usuario que no tenga conocimientos de informática se encontraría ante un botón del que probablemente no tendría ni idea de su uso, corriendo el riesgo de que lo averigüe justo al final de introducir todos los datos en el formulario, lo cual además, le supondrá un trabajo extra ya que ese tipo de personas suelen tener más dificultades en llenar formularios que los habituados al uso de internet y las herramientas informáticas.

Por ejemplo, a la hora de hacer una reclamación ante el Sistema Arbitral de Consumo la Junta de Andalucía ha creado una página web con un formulario para hacer reclamaciones:

La captura de pantalla muestra una ventana del navegador Firefox con la URL [www.consumoresponde.es/oficina\\_virtual/consumidores](http://www.consumoresponde.es/oficina_virtual/consumidores). La página tiene un encabezado con el logo de la Junta de Andalucía y el título "Oficina virtual | Consumo Responde". El formulario principal se titula "Escríbenos" y contiene los siguientes campos:

- Asunto:**
- Email:**
- Comentarios:**
- Nombre:**
- Apellido:**
- Provincia:**  - Elige Provincia -
- Segundo Apellido:**
- Municipio:**

Abajo del formulario, hay un botón "Subir fichero" con la descripción "No se ha seleccionado ningún archivo." y "Subir". Una nota indica: "Extensiones permitidas: txt doc pdf". A continuación, hay un enlace "Información Legal" y un botón "Enviar".

Captura de pantalla del formulario de reclamaciones ante el Sistema Arbitral de Consumo de la Junta de Andalucía en el navegador Mozilla Firefox.

Este formulario para una persona habituada al uso de internet será muy útil, ya que tiene pocos datos y es sencillo introducir el motivo de reclamación. Aunque incumple algunas máximas que se deberían evitar en los formularios, como por ejemplo que el botón de "Enviar" parece estar aparte del formulario. Sin embargo, una persona no habituada el uso de estas herramientas, posiblemente tendría algunas dificultades debido al orden de los datos y el que no se le indique que en los comentarios debe introducir el texto de la reclamación.

En todo caso, el formulario debe indicar a simple vista qué es lo que el usuario debe introducir en él. Y para ello no sólo hay que utilizar claridad en el texto

que va asociado a cada elemento, si no que incluso se pueden ligar estos usando la etiqueta LABEL.

A continuación vemos un ejemplo de petición de datos personales usando la claridad en los textos y dicha etiqueta:

```
</head>

<body>

<form action=>datos_personales.php</action method=>post</method>>

<table>

<tr><td><label for=>nombre</label>>Nombre:</label></td><td><input type=>text</type name=>nombre id=>nombre required /></td>
</tr><tr>

<td><label for=>apellidos</label>>Apellidos:</label></td><td><input type=>text name=>apellidos id=>apellidos required /></td>
</tr><tr>

<td><label for=>dni</label>>DNI:</label></td><td><input type=>text name=>dni id=>dni required /></td>
</tr><tr>

<td><label for=>domicilio</label>>Domicilio:</label></td><td><input type=>text name=>domicilio id=>domicilio /></td>
</tr><tr>

<td><label for=>poblacion</label>>Poblacion:</label></td><td><input type=>text name=>poblacion id=>poblacion /></td>
</tr><tr>

<td><label for=>cp</label>>Codigo Postal:</label></td><td><input type=>text name=>cp id=>cp /></td>
</tr><tr>
```

## UD1

```

<td><label for="email">Email: </label></td><td><input type="text" name="email" id="email" /> </td>

</tr><tr>

<td><label for="telefono">Telefono: </label></td><td><input type="text" name="telefono" id="telefono" /> </td>

</tr><tr>

<td></td><td><input type="submit" value="Enviar" /></td>
</tr>

</table>

</form>

</body>

</html>

```

The screenshot shows a Firefox browser window with the title "Ejemplo de formulario que tiene clarida...". The address bar displays "file:///ejemplo\_formulario\_claridad\_datos.html". The main content area contains a form with the following fields:

- Nombre:** [Empty input field]
- Apellidos:** [Empty input field]
- DNI:** [Empty input field]
- Domicilio:** [Empty input field]
- Poblacion:** [Empty input field]
- Codigo Postal:** [Empty input field]
- Email:** [Empty input field]
- Telefono:** [Empty input field]

At the bottom of the form is a blue "Enviar" button.

Captura de pantalla del resultado que debe dar el código al ejecutarlo en un navegador web.

#### 1.4.6. Utilización de páginas de error y de confirmación

Tanto en los formularios como en las páginas web en general la mayoría de usuarios habituales están acostumbrados a navegar a alta velocidad, apenas deteniéndose a consultar muchas de las acciones que hacen.

Esto es un problema a veces para los formularios, ya que instantáneamente un usuario puede darle al botón de enviar sin haber terminado realmente de completar la información, e inmediatamente arrepentirse de ello.

Cuando esto ocurre, en un formulario normal el usuario tendría que volver a restablecer el formulario, ya sea cargándolo originalmente o volviendo atrás en el navegador, perdiendo todos los datos que ha introducido.

Para evitar esto existen varias herramientas de confirmación del envío del formulario. Una simple ventana emergente, o incluso una página puente entre ambas sería más que suficiente para evitar que esto pasase.

Para hacerlo se puede hacer de dos formas:

- Programando en el lado del servidor una nueva página que recogería los datos enviados por el formulario y le pediría confirmación a este de si quiere realmente enviarlo. Si dice que sí, los datos recopilados se enviarían al servidor, si no, los datos recogidos se volverían a enviar al formulario, para volver a ponerlos en cada uno de sus campos adecuados.
- Programando un script en el cliente para que cuando el usuario le dé al botón de enviar le pregunte si desea hacerlo.

En ambos casos, el resultado es el mismo.



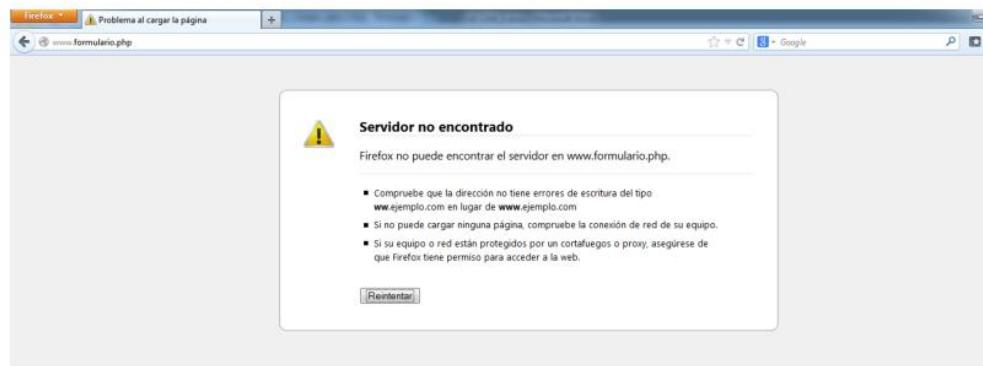
En la mayoría de formularios web el uso de páginas de confirmación no solo no es necesario, si no que es contraproducente, ya que entorpecerían la navegación y cansarían al usuario final. Por ejemplo el buscador de Google, si cada vez que un usuario fuese a hacer una búsqueda saliese una ventana de confirmación pidiendo al usuario si está seguro de hacerla, además de no servir para nada, puesto que si la búsqueda es incorrecta basta con repetirla cambiando los caracteres, el usuario podría cansarse de tener que confirmar cada vez que la realiza.

## UD1

Cuando un formulario envía información incorrecta o incompleta, a veces el navegador puede dar un mensaje de error. Esto se puede evitar con la programación correcta de los elementos, por ejemplo usando el atributo REQUIRED para que no se envíe el formulario con algunos campos que no están rellenos por el usuario.

Incluso puede ocurrir que el navegador haga una llamada a una página que no existe en esos momentos en el servidor. Pero esto debe ser programado en el mismo servidor, dejándose las páginas de error por defecto o enlazando a una página de error por defecto que se mostrará al usuario cada vez que se produzca un error intentando cargar una página web de dicho servidor.

Por ejemplo, esta es la página de error por defecto del navegador Mozilla Firefox:



Captura de pantalla de la página de error por defecto del navegador Mozilla Firefox.

Y en cambio, esta es la página de error por defecto de Google:



Captura de pantalla de la página de error de Google en el navegador Mozilla Firefox.

Las páginas de error deben ser programadas como si fuesen una página normal HTML, pero enlazadas a cualquier error que pueda encontrarse el usuario en la web. Esto se configura en el servidor en el fichero .htaccess, en la línea ErrorDocument 404. En caso de no configurar esto mostrará las páginas de error normales.

Por último, aquí tenemos un ejemplo del código en JavaScript que puede incluirse en una web para pedir confirmación del usuario cuando le da a enviar el formulario:

```
<html>

<head>

    <title>Ejemplo de confirmacion de envio de formulario.</title>

    <script language=>>JavaScript<>>></script>

        function confirmacion() {

            if (confirm('Pulsa Aceptar para enviar el formulario')) {

                {

                    document.formulario.submit()

                }

            }

        }

    </script>

</head>

<body>

    <form action="formulario.php" name="formulario">

        Nombre:

        <input type="text" name="nombre" value="" /> <br/>

        <input type="button" onclick="confirmacion()" value="Enviar" />

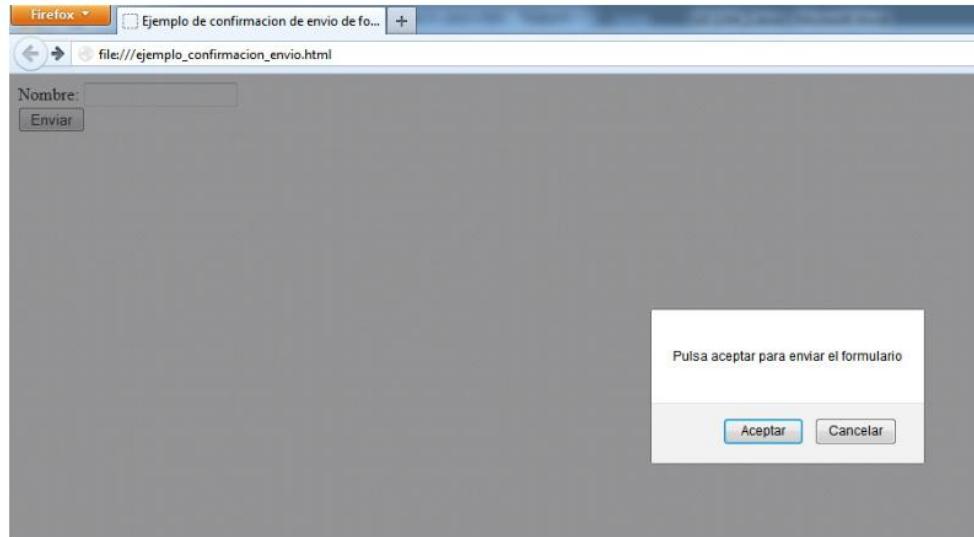
    </form>


```

## UD1

```
</body>
```

```
</html>
```



Captura de pantalla del resultado que debe dar el código al ejecutarlo en un navegador web.

Véase que el botón de envío del formulario en este caso no puede ser de tipo "submit", porque si no se ejecutaría el envío aunque se pulsase "cancelar". En este caso la función de submit está en la propia función JavaScript.

Área: informática y comunicaciones

# UD1

## Lo más importante

Tras finalizar esta Unidad Didáctica, el alumno debe ser capaz de dominar la creación de formularios web.

- En concreto, el alumno debe saber la utilidad de los formularios a la hora de añadir a las páginas web la interactividad necesaria para que los usuarios puedan introducir datos que se enviarán al servidor, debido a que normalmente las páginas web son meras herramientas estáticas, y con los formularios este factor se elimina.
- Además, el alumno puede en estos momentos identificar cómo se declara un formulario, y los atributos y elementos que se pueden encontrar en el mismo.
- Concretamente, podrá describir y definir la utilidad de los distintos campos y textos, sabiendo dónde incluirlos para la correcta funcionalidad del formulario.
- El alumno debe poder reconocer y utilizar las distintas etiquetas del mismo, no solo las más importantes como <FORM> o <INPUT>, si no incluso las menos comunes como <LABEL>, <LEGEND> o <TEXTAREA>.
- Además, el usuario deberá describir todos y cada uno de los elementos o controles de los formularios, empezando por los botones, diferenciando entre los botones de envío, los botones de borrado y el resto de botones.
- También deberá saber utilizar y explicar cuando deben utilizarse las listas desplegables con su etiqueta <SELECT>, así como todos sus atributos más importantes y explicar para qué se utilizan cada uno de ellos.

- El alumno deberá poder utilizar e identificar las casillas de verificación con el atributo CHECKBOX, así como saber las diferencias con los botones de opciones de atributo RADIobutton, conocer en qué situaciones pueden utilizar cada uno de ellos, y explicar y utilizar sus atributos más importantes cuando lo requiera.
- Además, el alumno podrá identificar los tres campos de texto principales que se utilizan en los formularios, como son los cuadros de texto de atributo TEXT, los cuadros de contraseña con el atributo PASSWORD y las áreas de texto con la etiqueta <TEXTAREA>. Deberá saber cuales son los atributos más relevantes que utilizan, y las diferencias entre ellos. Y sobre todo en qué situaciones es recomendable usar unos u otros.
- El alumno deberá tener claro cuál es la utilidad de agrupar los elementos del formulario mediante el uso de la etiqueta <FIELDSET>, así como nombrar a estas agrupaciones mediante el uso de la etiqueta <LEGEND>. Y además de trabajar con estas dos etiquetas, el alumno deberá hacer uso de sus atributos y de los beneficios que tiene poder aplicar modificaciones vía script a los elementos de dichas agrupaciones en lugar de tener que hacerlo de forma individual.
- También el alumno deberá identificar cuando es más adecuado usar varias páginas para programar un formulario, o cuando es mejor utilizar una solo.
- Deberá saber como mostrar al usuario final cuáles son los campos obligatorios, y como programarlos de forma que el usuario no pueda enviar el formulario sin que estos estén rellenos, con el atributo REQUIRED.
- El alumno deberá tener claro cuál es la ordenación lógica de los datos que se piden en un formulario, teniendo en cuenta el usuario final y, sobre todo, la claridad de los campos para que no haya errores en su inclusión.
- En esto último el alumno debe incluir la claridad de la información solicitada al usuario, porque si los datos que envía este son incorrectos, la finalidad del formulario sería inútil.
- Por último, y no por ello menos importante, el alumno debe saber la utilidad de las páginas de error, como modificarlas para que aparezcan las que se desea, y cómo incluir en el código del formulario una ventana de confirmación de envío. También deberá saber en que situaciones es recomendable programarlas y en que otras situaciones no lo es.

# UD1

## Autoevaluación

1. ¿Qué es una página web dinámica?
  - a. La que genera los contenidos en función de la hora del día.
  - b. La que genera los contenidos en función de los datos que introduce el servidor.
  - c. La que genera los contenidos en función de los datos que introduce el usuario.
  - d. La que genera los contenidos en función del tamaño de la pantalla.
2. ¿Qué es la etiqueta FORM?
  - a. Es la etiqueta que delimita y engloba a los elementos del formulario.
  - b. Es la etiqueta que se utiliza para dar forma al aspecto del formulario.
  - c. Es la etiqueta HTML que se utiliza para mostrar imágenes de extensión .form.
  - d. Es una etiqueta inexistente en HTML.

3. Cuál es la función del botón de envío creado usando el atributo INPUT con el tipo SUBMIT?
  - a. Indicar cuándo se debe enviar la información del formulario al servidor.
  - b. Borrar todos los datos del formulario.
  - c. Subir a la parte más superior del formulario.
  - d. No tiene ninguna función específica.
4. ¿Con qué etiqueta se declaran los elementos de una lista desplegable?
  - a. La etiqueta <SELECT>.
  - b. La etiqueta <OPTION>.
  - c. La etiqueta <OPTGROUP>.
  - d. La etiqueta <FORM>.
5. ¿Cuál es la gran diferencia entre las casillas de verificación y los botones de opción?
  - a. No existen diferencias entre ambas.
  - b. Permiten proporcionar varias opciones al usuario final.
  - c. Todas las opciones de las casillas de verificación son excluyentes entre sí.
  - d. Todas las opciones de los botones de opción son excluyentes entre sí.
6. ¿Cuál es la principal diferencia entre los cuadros de contraseña y los de texto?
  - a. Los campos de texto tienen varias líneas y los de contraseña solo una.
  - b. Los campos de texto y los campos de contraseña son iguales.
  - c. Los caracteres de los campos de contraseña no se ven a simple vista.
  - d. Los campos de contraseña son campos obligatorios en un formulario.

**UD1**

7. ¿Qué etiqueta es necesaria para nombrar a una agrupación de datos dentro de un formulario?
  - a. La etiqueta <FIELDSET>.
  - b. La etiqueta <LEGEND>.
  - c. La etiqueta <GROUP>.
  - d. La etiqueta <NAME>.
8. ¿Cuál es el atributo que se añade a los elementos de los formularios para que un campo deba ser rellenado obligatoriamente?
  - a. OBLIGATORY.
  - b. MANDATORY.
  - c. DEFAULT.
  - d. REQUIRED.
9. ¿Cuál es la máxima que debe cumplirse siempre que se programa cualquier tipo de formulario?
  - a. Se debe asumir que el usuario final no va a tener ningún conocimiento de informática.
  - b. Se debe programar siempre sin utilizar la etiqueta <FORM>.
  - c. Se deben poner siempre más campos de los necesarios para incluir más información.
  - d. Se debe evitar incluir, siempre que sea posible, un botón de tipo submit.
10. ¿De qué dos formas se pueden programar las páginas de confirmación de envío del formulario?
  - a. En lenguaje HTML y lenguaje BASIC.
  - b. Usando los atributos de ID y DISABLED.
  - c. En el lado del servidor y en el lado del cliente mediante un script.
  - d. Sólo se puede programar de una forma.

Área: informática y comunicaciones

# UD2

Plantillas en la  
construcción de  
páginas web

## UF1304: Elaboración de plantillas y formularios

## 2.1. Funciones y características

- 2.1.1. Descripción de una plantilla web
- 2.1.2. Elementos de una plantilla web
- 2.1.3. Estructura y organización de los elementos de las plantillas
- 2.1.4. Especificar las zonas modificables de una plantilla y las partes fijas
- 2.1.5. Utilización de plantillas

## 2.2. Campos editables y no editables

- 2.2.1. Definir y crear los campos susceptibles de cambios en una plantilla
- 2.2.2. Definir y crear los campos no modificables en una plantilla

## 2.3. Aplicar plantillas a páginas web

- 2.3.1. Las plantillas en la web
- 2.3.2. Búsqueda de plantillas en la red
- 2.3.3. Adaptación de plantillas a páginas web

## 2.1. Funciones y características

Una plantilla es, ni más ni menos, una herramienta para agilizar el trabajo de reproducción, de creación de muchas copias idénticas o casi idénticas de varios tipos de trabajos creativos.

Por lo tanto las plantillas son utilizadas habitualmente en la vida diaria, aún sin saberlo realmente. Por ejemplo, cuando se elabora una carta mentalmente siempre se sigue la misma plantilla de encabezado, cuerpo y pie de carta. Al igual que en la elaboración del currículum vitae, carta de presentación, invitaciones, etc.

El origen de las plantillas, o de las estructuras similares a las plantillas de hoy día, se remonta a la antigüedad de la civilización.

En el pasado hubo personas que se dieron cuenta que en ciertos trabajos, relacionados con el diseño y la construcción, el tener un modelo en el que basarse agilizaba mucho el trabajo de los artesanos.



Hay muchos estudios que sugieren que Leonardo da Vinci usaba la proporción áurea en sus obras, y que para comprobar la presencia de esa proporción en sus obras se utilizan unas plantillas geométricas que se superponen a dichas obras.

---

Estos modelos siguieron utilizándose y perfeccionándose con el paso de los años, hasta que se globalizó y popularizó la informática, que fue cuando entraron en escena las plantillas de programación.

En los inicios de la programación, existía una clara diferenciación entre los propios datos que manejaba el programa y sobre cómo se manipulaban. Se podría decir que los datos eran tipos muy simples, y los algoritmos eran diseñados para desarrollar funciones para trabajar con ellos.

El hecho de trabajar de esta forma dio lugar a que los programadores se dieron cuenta que gran parte del código que utilizaban, especialmente diversas funciones programadas para realizar funciones específicas, se repetían con bastante asiduidad en los programas.

Es más, algunos programas para realizar unas funciones específicas eran claramente muy parecidos entre sí, dando lugar a que resultase más rápido reutilizar el código que escribirlo de nuevo.

Por supuesto, reutilizar el código tiene dos problemas muy grandes:

- Las altas opciones de que se copien cosas que no deben utilizarse, creando código redundante que no es necesario.
- El que reutiliza el código debe tener, al menos, un conocimiento moderado del mismo, y si esto no es así debe estudiar su funcionamiento antes de reutilizarlo, por lo que realmente pierde mucha utilidad si el programador no es el mismo creador del código anterior.

Para esto se optó para una solución intermedia, que era la de poner el código que se iba a reutilizar en unas plantillas, que no son ni más ni menos que fragmentos de código que se especifica para que pueden utilizarse, y que además tienen una función específica global para la función especificada.

En las plantillas, además, el código tenía unas variables indeterminadas, que podían modificarse al gusto del programador. Y una gran cantidad de comentarios para indicar a los programadores que accedían por primera vez a ellas la utilidad de cada uno de los elementos de la misma.

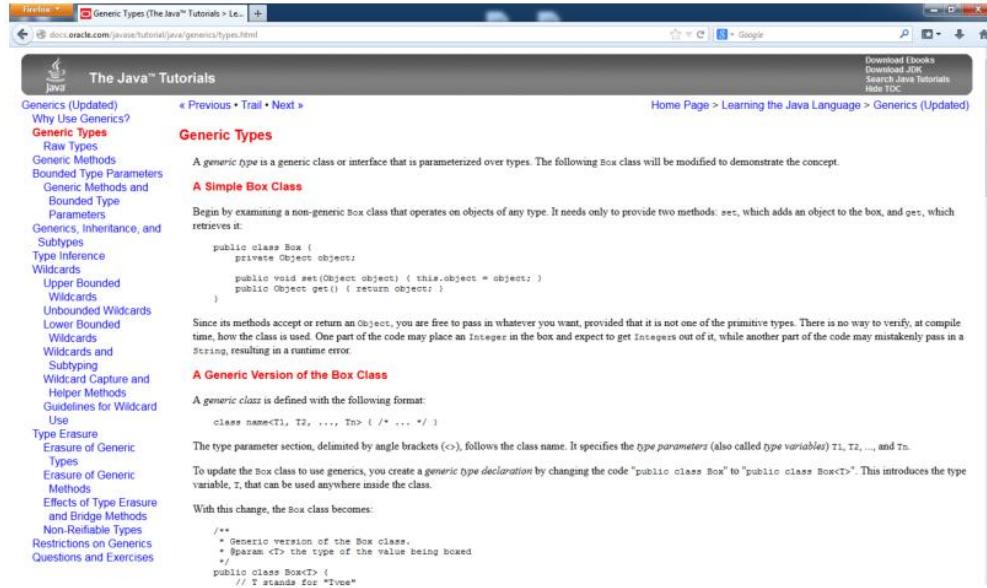
Más adelante apareció la programación orientada a objetos, la cual introdujo nuevas opciones a la hora de programar, y por supuesto, muchas más facilidades a la hora de hacer las plantillas, ya que al poder crear tipos más complejos, a su vez podían hacerse plantillas de estos de forma mucho más sencilla y eficaz.

En dicha programación, de hecho, se pudieron crear plantillas para definir una clase o una función. Posibilitando esto, al crear funciones o clases genéricas, que hacer uso de plantillas fuese algo, no solo recomendable, si no habitual a la hora de programar.

## UD2

En esta última década con la popularización de internet, la proliferación de plataformas con plantillas para todo tipo de lenguajes ha hecho que, para los programadores, la vida sea mucho más sencilla, siempre que sepas lo que necesitas y donde buscar.

Por ejemplo, en la página web de oracle cualquiera puede usar, de forma gratuita, varias plantillas para java:



Captura de pantalla de diversas plantillas de java en la página web de Oracle en el navegador Mozilla Firefox.

La proliferación de internet, y el hecho de que los lenguajes de programación estén en constante evolución, hacen que el uso de plantillas sea, hoy por hoy, casi un obligado en la programación, no sólo para ahorrar tiempo, sino también para facilitar el trabajo a los programadores.

Esto ocurre con todos los lenguajes de programación más utilizados en la actualidad, que son los siguientes:

- C.
- C++.
- Java.
- Python.
- PHP.

- SQL.
- HTML.
- JavaScript.
- XML.
- Perl.
- AJAX.
- Ruby.

Incluso se ha convertido en una oportunidad de negocio, ya que hay personas que se dedican en exclusiva a crear plantillas para venderlas en páginas web. Por supuesto, estamos hablando ya de plantillas más avanzadas para diversos Sistemas de Gestión de Contenidos, o como son llamados en inglés CMS (Content Management System).

### 2.1.1. Descripción de una plantilla web

Las plantillas web no se diferencian, en principio, de una plantilla de programación normal, salvando las diferencias entre los propios lenguajes, claro.

Las plantillas para los lenguajes HTML, JavaScript o PHP son, ni más ni menos, que fragmentos de código que tiene diversas utilidades web, con variables estándar y con comentarios para que el programador sepa cómo implementarlos en la web.

Realmente una plantilla web es, cuando un programador crea y personaliza un sitio web con la intención de crear una plantilla y, a continuación, guarda el sitio web como plantilla, creándose así una plantilla web. Las plantillas web deben almacenarse de forma que sean fácilmente localizables, por ejemplo en un nivel superior de donde se encuentra el sitio o los sitios webs a crear, para que esté disponible para la creación de subsitios utilizándola.

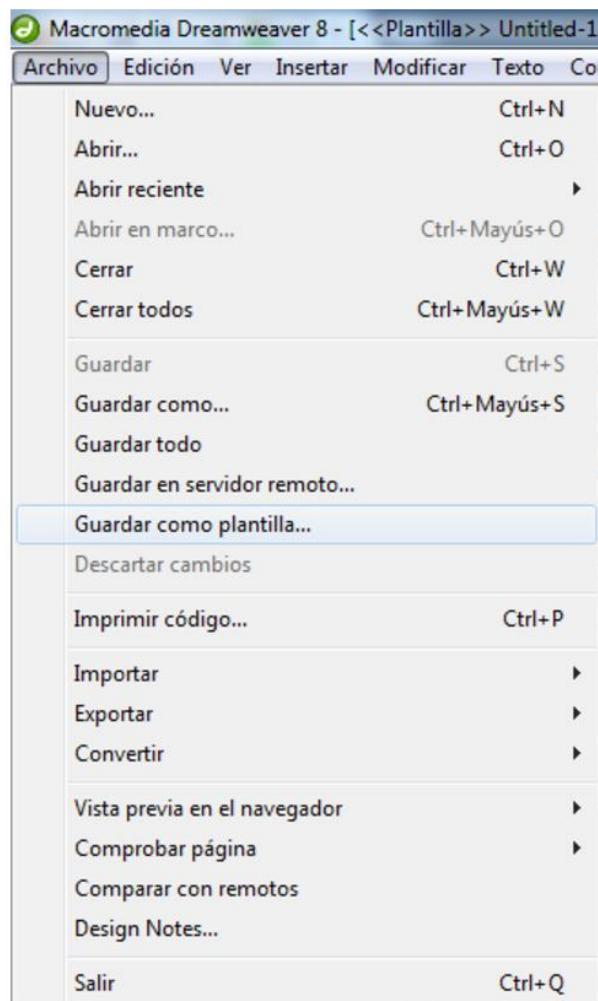
Es recomendable, además, almacenar las plantillas web con una extensión distinta al archivo original, para que no haya confusiones entre una plantilla y un documento web. Por ejemplo, si se crean y almacenan plantillas usando la herramienta Dreamweaver, las plantillas se almacenan con la extensión dwt, pero se puede elegir cualquier extensión que sirva para este fin, siempre que no se confundan con las extensiones originales.

## UD2



Existen aplicaciones destinadas a la creación, construcción, diseño y edición de páginas web. Las más conocidas son Adobe Dreamweaver, Microsoft Expression Web y BlueGriffon, que es de código abierto. Estos programas son, por sí, herramientas con plantillas web para facilitar el trabajo al programador y que pueda observar el proceso de su creación de forma visual cuando está creando el código.

A continuación vemos como se guarda una plantilla usando Dreamweaver:



## UF1304: Elaboración de plantillas y formularios

Captura de pantalla de la opción para guardar un documento HTML como plantilla en Dreamweaver 8.

Por supuesto, las plantillas web siempre se deben almacenar preferentemente en sitios web, desde donde poder acceder a sus distintos elementos.

Proceso de creación de una plantilla web:

- Identificación del elemento que se quiere crear una plantilla.
- Creación del código de uno de estos elementos.
- Hacer que las variables de este elemento sean lo más genéricas posible.
- Poner comentarios indicando lo que forma parte de la plantilla y qué tiene que incluir el programador cuando la aplique.
- Guardar la plantilla con una extensión distinta a la de un archivo normal en ese lenguaje, además si se está trabajando en un sitio web, almacenarla en un sitio especial, por encima de la raíz.

Por supuesto, no es lo mismo una plantilla para un elemento común de HTML, como pueda ser para un simple botón que haga una función muy simple, que una plantilla que modifique el diseño completo de la presentación web. Este último tipo de plantillas se pueden, de hecho, aplicar sobre páginas aunque ya se haya comenzado su diseño, puesto que en realidad no sobrescriben el código, si no que lo modifican y mejoran para cambiar la web. Por ejemplo, si tenemos la web creada mediante el CMS wordpress, podríamos elegir como aspecto básico la siguiente plantilla:

**WordPress 3.8.1 Maintenance Release**

Posted January 23, 2014 by Andrew Nacin. Filed under Releases.

After six weeks and more than 9.3 million downloads of WordPress 3.8, we're pleased to announce WordPress 3.8.1 is now available.

Version 3.8.1 is a maintenance release that addresses 31 bugs in 3.8, including various fixes and improvements for the new dashboard design and new themes admin screen. An issue with taxonomy queries in WP\_Query was resolved. And if you've been frustrated by submit buttons that won't do anything when you click on them (or thought you were going crazy, like some of us), we've found and fixed this "dead zone" on submit buttons.

It also contains a fix for embedding tweets (by placing the URL to the tweet on its own line), which was broken due to a recent Twitter API change. (For more on Embeds, see the Codex.)

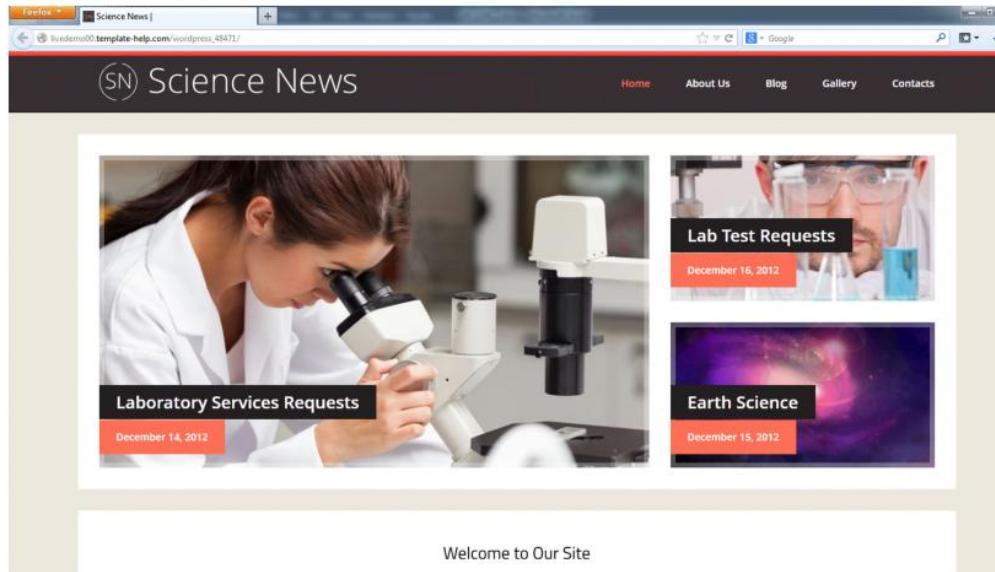
For a full list of changes, consult the [list of tickets](#) and the [changelog](#). There's also a detailed summary for developers on the development blog.

If you are one of the millions already running WordPress 3.8, we will start rolling out automatic background updates for WordPress 3.8.1 in the next few hours. For sites

## UD2

Captura de pantalla del blog básico de Wordpress en el navegador Mozilla Firefox.

Sin embargo, una web creada mediante ese mismo CMS, Wordpress, y al que se le ha aplicado una plantilla y un tema concreto podría tener este aspecto:



Captura de la plantilla de un tema de Wordpress en el navegador Mozilla Firefox.

En todo caso, cuando se crea una plantilla web, sea esta de un pequeño elemento como de un botón, o de toda una página completa, esta deberá contar con elementos comunes.

Por ejemplo, este es un ejemplo de una plantilla HTML de la creación de un formulario de ingreso a una página web:

```
<!-- Comienzo de la plantilla -->

<html>

<head>

    <title><!-- Introducir aquí el título de la página --></title>

</head>

<body>
```

```

<form action="!-- Web de destino -->" method="post">

    Usuario:

        <input type="text" size="30" maxlength="30"
name="usuario" value="Usuario" />

        <br/>

    Password:

        <input type="password" size="10" maxlength="10"
name="password" />

        <br/>

        <input type="submit" value="Entrar" />

    </form>

</body>

</html>

<!-- Fin de la plantilla -->

```

### 2.1.2. Elementos de una plantilla web

Las preguntas que debe hacerse el programador cuando quiere diseñar una plantilla web son las siguientes:

- ¿En qué lenguaje quiere hacer su plantilla web?
- ¿En qué situaciones necesitará las plantillas web?
- ¿Necesitará la plantilla web para usarla de forma habitual o la usará de forma esporádica?
- ¿Realmente necesita una plantilla para ese elemento o no lo usará tan a menudo como para necesitar una plantilla?
- Y sobre todo, ¿ganará tiempo de trabajo diseñando la plantilla o realmente no merece la pena porque le supone una pérdida de tiempo?

## UD2



El saber si el crear una plantilla va a ahorrar tiempo, y si el elemento sobre el que se hace la plantilla va a ser reutilizado, es algo muy importante, ya que realmente las plantillas están diseñadas para hacer ganar tiempo al programador, y si no tienen esa función, no merece la pena realizar una plantilla web.

---

A partir de esas preguntas, el programador puede diseñar un boceto inicial en código de lo que pretende que sea su plantilla. A partir de ahí creará los elementos necesarios para crear un programa que haga lo que necesite para su plantilla.

A continuación tendrá que dividir los elementos en dos tipos:

- Campos modificables.
- Campos no modificables.

Como especificar estos elementos se explicará más adelante en esta Unidad Didáctica, pero es importante que el alumno sepa que existen estos dos tipos de elementos. Además, en las plantillas web, especialmente en las plantillas HTML, habrá una serie de elementos que deben ser fijos por si se reutiliza el código en su totalidad. Estos son:

- Las etiquetas de apertura y cierre del código HTML.
- Las etiquetas de apertura y cierre de la cabecera, junto con información relacionada con la plantilla.
- Las etiquetas de apertura y cierre del cuerpo del código, dónde irá el contenido en sí de nuestra plantilla HTML.

En todo caso, incluyendo las plantillas web en otro tipo de lenguajes como JavaScript o PHP, es importante que el programador indique correctamente, mediante comentarios, la función de cada elemento en la plantilla.

La inclusión de comentarios en el código se hará de la forma habitual:

- Mediante esta fórmula en HTML: `<!--El comentario va aquí-->`
- Mediante esta fórmula en JavaScript y PHP: `/*El comentario va aquí*/`

### 2.1.3. Estructura y organización de los elementos de las plantillas

Estos elementos explicados en el punto anterior, que son en especial:

- El Código fijo que lleva toda plantilla dependiendo del lenguaje.
- Los elementos o campos modificables.
- Y los elementos o campos no modificables.

Deben ser debidamente organizados y estructurados.

Es decir, lo primero que debe hacerse es realizar un código lo más correcto y elegante posible, con tabulaciones y referencias a aperturas y cierres de forma clara y precisa.

Una vez hecho esto, la identificación siempre de los distintos elementos de los comentarios siguiente siempre el mismo protocolo de comentarios es lo más correcto y útil a la hora de poder consultar decenas o incluso cientos de plantillas en un sitio web concreto.



Todos los elementos modificables de la plantilla, como ya veremos más adelante, deben ser nombrados de forma que se sepa claramente qué función cumplen en la plantilla. Por ejemplo, si hacemos una plantilla de un formulario que recopila los datos personales del usuario, para ser incluida en la página web sin tener que programarla, es muy importante que los elementos que puedan variar, como por ejemplo la página a la que va a enviar los datos el formulario cuando se pulse submit, sea un valor lo más claro posible, como poner ejemplo pagina\_destino.php.

---

Por ejemplo, veamos una plantilla por si se quiere incluir el formulario de un buscador en una página web en HTML:

```
<!-- Comienzo de la plantilla -->
<html>
```

## UD2

```

<head>
    <title>Ejemplo de buscador.</title>
</head>
<body>
    <form action="pagina_destino.php<!-- Introducir aquí
la página de destino donde se van a enviar los datos cuan-
do se pulse submit -->" method="post">
        Introducir datos a buscar:
        <br/>
        <input type="text" name="buscador" />
        <br/><br/>
        <input type="submit" value="Buscar"/>
    </form>
</body>
</html>
<!-- Fin de la plantilla -->

```

### 2.1.4. Especificar las zonas modificables de una plantilla y las partes fijas

En toda plantilla web, habrá partes que necesiten modificarse, y partes que no deban tocarse. Es más, hay plantillas que no deben modificarse en ningún momento.

El por qué hay plantillas que no deben modificarse puede ser por varias razones:

- El código es fijo, como cuando se crea un pequeño efecto de presentación mediante el uso de JavaScript.
- Los efectos del código son desconocidos porque los ha hecho un tercer programador. En estos casos a veces es recomendable no tocar el código si no es necesario.

En ambos casos, se utilizarán los comentarios si esto se deja así, para guiar a un futuro programador o al mismo programador actual, en caso de una revisión del código futura.



Siempre es recomendable no usar código de terceras personas siempre que sea posible, pero a veces esto no tiene sentido, ya que hay comunidades abiertas en internet de donde bajarse plantillas para utilizarlos en los programas de forma gratuita, así que no compensa estar horas haciendo algo que ya ha sido diseñado por una persona. Eso sí, aunque existan estos pedazos de código ya hechos, eso no quita que el programador deba saber cómo incluirlos, y cómo interacciona con el funcionamiento de su propia web.

Las zonas de una plantilla que sí puedan y deban ser modificadas deberán estar debidamente indicadas por el programador que ha realizado la plantilla, y el programador deberá modificarlas para incluirlas en su código.

Tras hacer esto, el programador debe eliminar todo comentario relacionado con estos campos modificables que se corra el riesgo a que puedan dar lugar a equívocos.

Por ejemplo en el código de la pantalla anterior:

```
<form action="pagina_destino.php<!-- Introducir aquí  
la página de destino donde se van a enviar los datos cuan-  
do se pulse submit -->" method="post">
```

Si utilizamos esta plantilla web y la modificamos para incluirla en nuestro programa, debemos tener en cuenta que:

- Debemos cambiar pagina\_destino.php por la auténtica página a la que se quieran enviar los datos en el servidor.
- El comentario <!-- Introducir aquí la página de destino donde se van a enviar los datos cuando se pulse submit --> debe ser eliminado.

El resultado final sería el siguiente:

```
<form  
method="post"> action="pagina_de_destino_real.php"
```

## UD2

Que como se puede observar es mucho más corto, y además mucho más fácil de visualizar, que la anterior línea de código con el comentario.



Importante

El hecho de eliminar comentarios no implica que el programador deba dejar de usar comentarios propios cuando la situación así lo requiera, que metodológicamente debería ser cuantas más veces posible, sin llegar a pasarse.

---

Veamos un ejemplo de una plantilla con elementos JavaScript con zonas modificables y con zonas no modificables:

```
<!-- Comienzo de la plantilla -->

<html>

<head>

    <title>Ejemplo de plantilla JavaScript.</title>

    <script language="JavaScript">

        function abrir_ventana (pagina)

        {

            var      opciones="toolbar=no,           location=no,
directories=no,   status=no,   menubar=no,   scrollbars=no,
resizable=yes, width=508, height=365, top=85, left=140";

            window.open(pagina,"",opciones);

        }

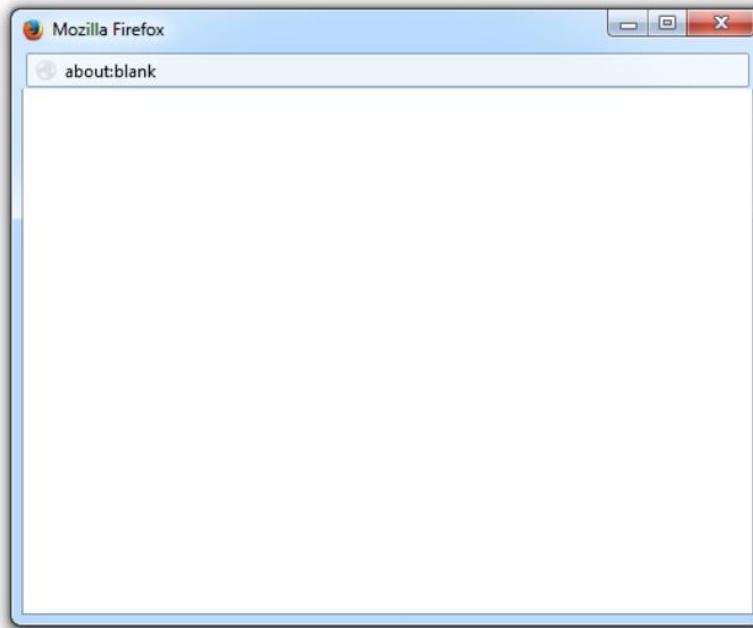
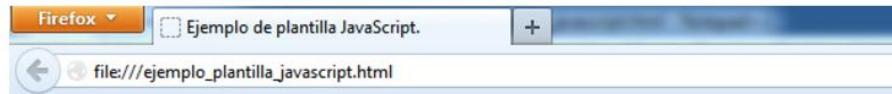
    </script>

</head>

<body onload="abrir_ventana('ventana_emergente.html<!--
Este elemento se deberá sustituir por la url que se quiere
que se muestre en la ventana emergente que se abrirá al
cargar la página -->')">
```

## UF1304: Elaboración de plantillas y formularios

```
</body>  
</html>  
!-- Fin de la plantilla -->
```



Captura de pantalla del resultado que debe dar el código al ejecutarlo en un navegador web.

Como el alumno puede comprobar, la mayoría del código en principio no debería ser modificado, excepto la página que se va a cargar en la ventana emergente cuando se abre la página web, y el comentario asociado.

Sin embargo, un programador familiarizado con JavaScript podría, si lo quisiese, modificar las llamadas a la función y la función en sí, por ejemplo. Pudiendo así, modificar el programa a su gusto.

En todo caso, es muy importante que se puedan identificar fácilmente en una plantilla web cuales son los elementos modificables y cuáles no.

## UD2

### 2.1.5. Utilización de plantillas

Las plantillas son unas herramientas muy poderosas cuando se utilizan correctamente, ya que no solo permiten al programador ahorrar mucho tiempo de trabajo, maximizando así su productividad, si no que también permiten hacer que la inspiración de un día sea útil en el futuro en muchos otros días de trabajo.

Pero a pesar de eso, no siempre es recomendable utilizar plantillas, y el programador debe medir cuando es necesario usarlas y cuando no.

Cuando se va a construir una plantilla web o utilizar una plantilla ya sea propia o de otro programador se debe pensar en los siguientes factores:

- ¿Es un elemento tan sumamente complejo como que para merezca la pena la inversión en tiempo de diseñar una plantilla o de buscarla?
- Por lo tanto, si se usa, ¿se va a ganar tiempo utilizándola o en cambio se va a perder más del que se usaría programándola normalmente?
- Si se usa una plantilla web, ¿sería la solución más elegante ante el problema que tenemos planteado o en cambio sería más recomendable desarrollar el código nosotros mismos?
- Si vamos a usar una plantilla de pago, ¿podemos nosotros o nuestro empleador asumir el coste de una plantilla de pago?

Si tras hacerse estas preguntas, aún se piensa que es recomendable usar una plantilla web, se debe proceder a crearla o a utilizar una ya creada.



Este tipo de preguntas con la experiencia se resuelven con mucha más velocidad, casi inconscientemente, ya que el programador ya se da cuenta de inmediato cuando necesita o no usar una plantilla web. Sin embargo, es recomendable hacerse estas preguntas tranquilamente cuando se tiene menos experiencia para poder asumir cuando es necesario usar plantillas o no, y sobre todo, para ahorrar mucho trabajo en el futuro usándolas.

Uno de los grandes problemas con los que se suele encontrar el trabajo haciendo programas, ya sea programas web u otro tipo de programadores, es el enfrentarse a la página en blanco.

Cuando se tiene que trabajar varias semanas con un programa en concreto, el programador está casi siempre trabajando con muchas cientos, o incluso miles de líneas de código, pero cuando acaba de trabajar con ese programa y empieza uno nuevo, el tener que enfrentarse a la página en blanco y empezar de cero supone un reto.

Para ello, las plantillas también son las amigas del programador.

Por ejemplo, si se trabaja diseñando webs en HTML, una simple plantilla creada por el propio programador para sí mismo de la estructura básica de un código HTML podría evitar tener que enfrentarse a esa temida línea en blanco.

Pero no sólo eso. Si se quiere hacer una tabla en HTML, comprobar la estructura básica de párrafos en HTML, hacer listas en HTML, crear enlaces o imágenes en HTML, construir tablas o formularios en HTML, o incluso crear clases propias CSS, todo ello es susceptible de ser creado con una plantilla, como veremos a continuación.

Cuando un programador se enfrenta a una página en blanco, muchas veces el tener una estructura básica de HTML en una plantilla para que sirva como inicio puede ser realmente útil.

Muchos de estos elementos podrían, incluso, no acabar sirviendo en el diseño de la página web, pero simplemente están pensados para servir de información indirecta al programador de dónde tiene que ir poniendo los fragmentos de código que cree para esa página.

Veamos un ejemplo de plantilla de estructura básica de HTML:

```
<!-- Comienzo de la plantilla -->

<html>

<head>

<title>Plantilla de estructura basica de HTML.</title>

</head>

<body>

<h1>Esta es la cabecera principal</h1>
```

## UD2

```

<p>Esta es la introduccion del resto de la pagina. Si esta es muy larga podria estar dividida en subcabeceras.</p>

<h2>Esta es la subcabecera</h2>

<p>Aqui ira el cuerpo del contenido bajo la subcabecera</p>

<h2>Otra subcabecera</h2>

<p>Y aqui va otro contenido de la subcabecera.</p>

</body>

</html>

<!-- Fin de la plantilla -->

```



## Esta es la cabecera principal

Esta es la introducción del resto de la página. Si esta es muy larga podría estar dividida en subcabeceras.

### Esta es la subcabecera

Aquí irá el cuerpo del contenido bajo la subcabecera

### Otra subcabecera

Y aquí va otro contenido de la subcabecera.

Captura de pantalla del resultado que debe dar el código al ejecutarlo en un navegador web.

Cuando un programador crea una nueva página HTML puede que ni siquiera use esa misma estructura de párrafos, pero aunque no utilice esos códigos, de un simple vistazo puede ver dónde colocar los elementos principales a modo de cabecera, y dónde colocar los subelementos de la página web, más debajo de este.

Además, en este tipo de plantillas la gran ventaja que tiene es que no necesita demasiados comentarios para indicarles a futuros programadores qué es cada cosa, porque el propio código informa.

A veces cuando se está programando en HTML se quiere crear una estructura de párrafos en dicho lenguaje. Y para ello hay una serie de etiquetas preparadas para hacerlo.

La mayoría de programadores habituales de HTML se saben estas etiquetas de memoria, sólo por su uso, por lo que no tiene sentido tener una plantilla sobre ellas.

Pero a veces, ya sea porque el programador es un recién iniciado en este lenguaje, o simplemente porque no use demasiado estas etiquetas, el tener que consultar cómo construir una estructura de párrafos gasta un tiempo que bien podría ser ahorrar si se usase una plantilla.

Veamos un ejemplo de una plantilla de la estructura de los párrafos en HTML:

```
<!-- Comienzo de la plantilla -->

<html>

<head>

    <title>Plantilla de estructura de parrafos de HTML.</
title>

</head>

<body>

    <!-- Estructura en parrafos -->

    <h1>Esta es la cabecera principal</h1>

    <h2>Esta es la cabecera de nivel 2</h2>

    <h3>Esta es la cabecera de nivel 3</h3>

    <h4>Esta es la cabecera de nivel 4</h4>

    <h5>Esta es la cabecera de nivel 5</h5>

    <h6>Esta es la cabecera de nivel 6</h6>

    <p>Este es el texto en parrafos.</p>

    <!-- Ejemplo de utilidad de los parrafos y texto -->

    <h1>Titulo de un libro</h1>
```

## UD2

```

<h2>Capítulo 1</h2>

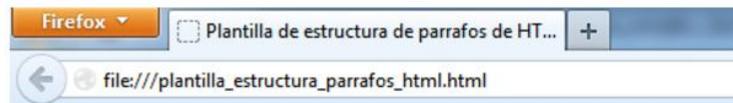
<p>Esto es texto normal, <i>en cursiva, </i> <b>en negrita </b> y <u>subrayado</u>. </p>

</body>

</html>

<!-- Fin de la plantilla -->

```



# Esta es la cabecera principal

## Esta es la cabecera de nivel 2

### Esta es la cabecera de nivel 3

#### Esta es la cabecera de nivel 4

##### Esta es la cabecera de nivel 5

###### Esta es la cabecera de nivel 6

Este es el texto en párrafos.

# Titulo de un libro

## Capítulo 1

Esto es texto normal, *en cursiva*, **en negrita** y subrayado.

Captura de pantalla del resultado que debe dar el código al ejecutarlo en un navegador web.

A veces, cuando se hacen páginas web, es necesario ordenar diversos elementos en listas. Para ello el propio HTML cuenta con herramientas adecuadas para ello.

Sin embargo, estas estructuras realmente se utilizan muy poco, e incluso los programadores más experimentados puede que tengan dificultades en algún momento para recordar la estructura de etiquetas de forma exacta.

Para ello están las plantillas. Y a continuación tenemos el ejemplo de una plantilla donde se muestran listas no ordenadas y ordenadas:

```
<!-- Comienzo de la plantilla -->

<html>

  <head>
    <title>Plantilla de estructura de listas de HTML.</title>
  </head>

  <body>

    <h1>Listas en HTML</h1>

    <!-- Listas no ordenadas -->
    <h2>Listas no ordenadas</h2>
    <ul>
      <li>No ordenada 1</li>
      <li>No ordenada 2</li>
      <li>No ordenada 3</li>
    </ul>

    <!-- Listas ordenadas -->
    <h2>Listas ordenadas</h2>
    <ol>
      <li>Ordenada 1</li>
      <li>Ordenada 2</li>
      <li>Ordenada 3</li>
    </ol>
  </body>
</html>
```

## UD2

```

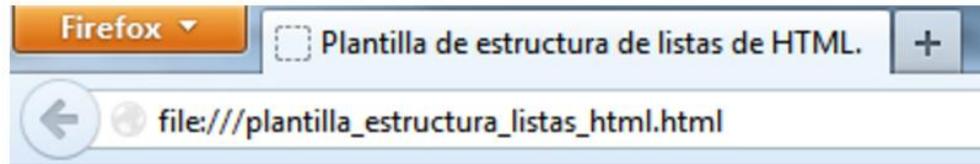
</ol>

</body>

</html>

<!-- Fin de la plantilla -->

```



# Listas en HTML

## Listas no ordenadas

- No ordenada 1
- No ordenada 2
- No ordenada 3

## Listas ordenadas

1. Ordenada 1
2. Ordenada 2
3. Ordenada 3

Captura de pantalla del resultado que debe dar el código al ejecutarlo en un navegador web.

En la programación web uno de los elementos más importantes, sin duda alguna, son los enlaces a otras páginas web. Gracias a estos enlaces el usuario puede navegar de un sitio a otro, lo que es a su vez la base de la World Wide Web.

Cómo incluir estos enlaces en HTML puede ser bastante sencillo, pero muchas veces estos no se incluyen por defecto en la mayoría de programas web. E incluso hay algunos tipos de estos enlaces con los que no se trabaja de forma habitual, como por ejemplo, el enlace que se incluye para que el usuario final incluya un correo.

Para que el programador tenga una base sobre ella se puede utilizar una plantilla web como la siguiente:

```
<!-- Comienzo de la plantilla -->

<html>

<head>

    <title>Plantilla de estructura de URLs de HTML.</title>

</head>

<body>

    <h1 id="top">Estructura de las URL en HTML</h1>

    <h2>URL normal con texto</h2>

    <p>
        <!-- Estructura para crear una URL -->
        <a href="url_con_texto.com">Enlace URL</a>
    </p>

    <h2>URL para enviar correo</h2>

    <p>
        <!-- Estructura para crear una URL a un correo -->
        <a href="mailto:ejemplo@correo.com">Enviar correo</a>
    </p>

    <h2>URL para ir a la parte superior de la pagina</h2>

    <p>
```

## UD2

```
<!-- Estructura para crear una URL a un ancla de la
web, en este caso a la parte superior-->
```

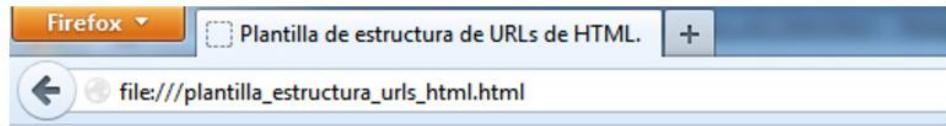
```
<a href="#top">Ir a parte superior</a>
```

```
</p>
```

```
</body>
```

```
</html>
```

```
<!-- Fin de la plantilla -->
```



# Estructura de las URL en HTML

## URL normal con texto

[Enlace URL](#)

## URL para enviar correo

[Enviar correo](#)

## URL para ir a la parte superior de la pagina

[Ir a parte superior](#)

Captura de pantalla del resultado que debe dar el código al ejecutarlo en un navegador web.

Al igual que las URLs, otro de los elementos que definen la popularidad de las páginas web en las últimas dos décadas han sido la inclusión de imágenes en ellas.

Cómo incluir estas imágenes en HTML también puede ser considerado bastante sencillo. De hecho hoy en día es prácticamente imposible encontrar una página web que no incluya imágenes. Pero a pesar de ello el código para incluirlas en HTML no siempre tiene porqué tenerse claro.

Así que, lo mejor en estos casos es crear una plantilla web como la siguiente:

```
<!-- Comienzo de la plantilla -->

<html>

<head>

    <title>Plantilla de estructura de inclusión de imágenes en HTML.</title>

</head>

<body>

    <h1>

        <!-- Enlace a una imagen normal -->

        <!-- El valor de "alt" muestra un texto alternativo si no se muestra la imagen -->

        <!-- Los valores de "height" y "width" representan la altura y anchura de la imagen -->

    </h1>

    <!-- Enlace a una imagen encuadrada -->

    <figure>

        <p>

            <!-- El contenido de "figcaption" es el texto que se va a mostrar en el pie de la imagen -->

            <figcaption> Este es el texto que se muestra bajo la imagen</figcaption>

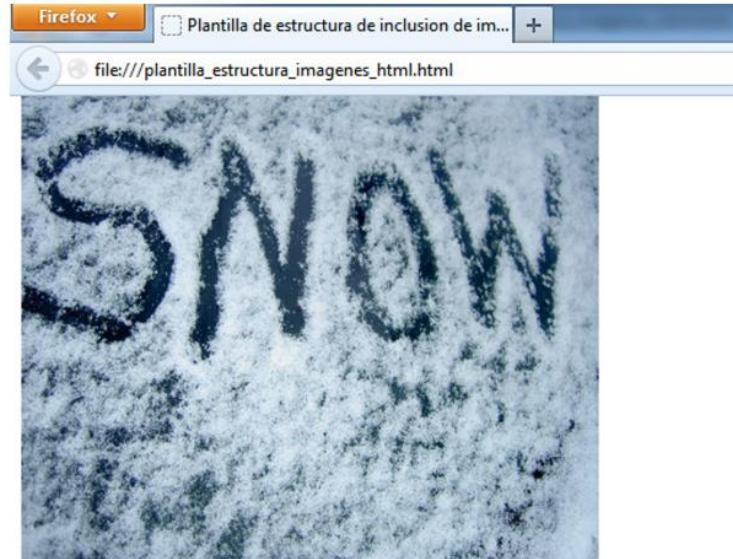
        </p>

    </figure>


```

## UD2

```
</body>  
</html>  
<!-- Fin de la plantilla -->
```



Este es el texto que se muestra bajo la imagen

Captura de pantalla del resultado que debe dar el código al ejecutarlo en un navegador web.

Para organizar y distribuir los distintos elementos en una página web programada en HTML se pueden utilizar muchos sistemas. Uno de ellos es mediante tablas en HTML.

Por supuesto, en un momento dado también puede que sea necesario usar una tabla para mostrar resultados concretos, como por ejemplo los valores de una estadística, o simplemente unos números o listados cualesquiera.

En todo caso, para facilitar la inclusión de estas tablas en el código HTML, se puede crear una plantilla con este fin con la estructura de las tablas en este lenguaje.

Veamos un ejemplo de esta plantilla:

```
<!-- Comienzo de la plantilla -->

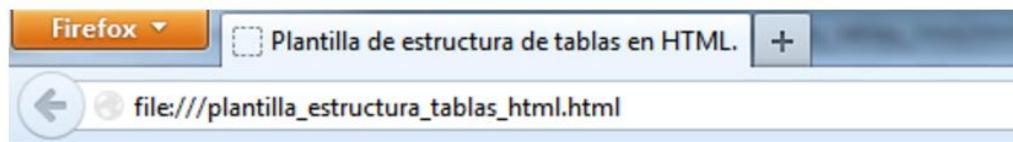
<html>
  <head>
    <title>Plantilla de estructura de tablas en HTML.</title>
  </head>
  <body>
    <table>
      <thead>
        <tr><th></th><th scope="col">Cabecera columna 1</th><th scope="col">Cabecera columna 2</th></tr>
      </thead>
      <tbody>
        <tr><th scope="row">Cabecera linea 1</th><td>Casilla linea 1 columna 1</td><td>Casilla linea 1 columna 2</td></tr>
        <tr><th scope="row">Cabecera linea 2</th><td>Casilla linea 2 columna 1</td><td>Casilla linea 2 columna 2</td></tr>
      </tbody>
    </table>
  </body>
</html>
```

## UD2

```

<tfoot>
<tr><td></td><td colspan="2">Pie de tabla</td></tr>
</tfoot>
</table>
</body>
</html>

<!-- Fin de la plantilla -->
```



**Cabecera columna 1    Cabecera columna 2**  
**Cabecera linea 1** Casilla linea 1 columna 1 Casilla linea 1 columna 2  
**Cabecera linea 2** Casilla linea 2 columna 1 Casilla linea 2 columna 1  
 Pie de tabla

Captura de pantalla del resultado que debe dar el código al ejecutarlo en un navegador web.

Uno de los elementos más importante, sin duda alguna, del lenguaje HTML son los formularios. De hecho, en las páginas webs actuales es muy infrecuente que no exista, al menos, un formulario de búsqueda en su código.

Veamos un ejemplo de una plantilla de un formulario con varios elementos de su código:

```

<!-- Comienzo de la plantilla -->
<html>
<head>
    <title>Plantilla de estructura de formularios en HTML.</title>
</head>
```

## UF1304: Elaboración de plantillas y formularios

```
<body>

    <form action="pagina_web_de_destino" method="get">

        <fieldset>

            <legend>
                Agrupacion de elementos:
            </legend>

            <label>
                Texto plano:
                <input type="text" name="texto_plano" size="30" maxlength="100">
            </label>

            <br />

            <label>
                Email:
                <input type="email" name="email" size="30" maxleng-
th="100">
            </label>

            <br />

        </fieldset>

        <br />

        <fieldset>

            <legend>
                Otra agrupacion de elementos:
            </legend>

            <p>
                <label for="lista">
```

## UD2

```

Lista desplegable

</label>

<select name="lista" id="lista">

    <option value="opcion1">Opcion 1</option>
    <option value="opcion2">Opcion 2</option>
    <option value="opcion3">Opcion 3</option>

</select>

</p>

<p>

Botones de opcion:

<br />

<label>

<input type="radio" name="opciones" value="opcion1" />

Opcion 1

</label>

<label>

<input type="radio" name="opciones" value="opcion2" />

Opcion 2

</label>

<label>

</p>

<p>

<label for="area_texto">

Area de texto:

</label>

```

## UF1304: Elaboración de plantillas y formularios

```
<br />

<textarea rows="4" cols="40" id="area_texto">
</textarea>

</p>

<label>

<input type="checkbox" name="casilla" checked="marcada"
/>

    Casilla de verificacion

</label>

<br />

<input type="submit" value="Boton de Enviar" />

</fieldset>

</form>

</body>

</html>

<!-- Fin de la plantilla -->
```

## UD2

**Firefox** ▾ Plantilla de estructura de formularios en ... +

file:///plantilla\_estructura\_formularios\_html.html

Agrupacion de elementos:

Texto plano: \_\_\_\_\_

Email: \_\_\_\_\_

---

Otra agrupacion de elementos:

Lista desplegable Opcion 1 ▾

Botones de opcion:

Opcion 1  Opcion 2

Area de texto:

Casilla de verificacion

**Boton de Enviar**

Captura de pantalla del resultado que debe dar el código al ejecutarlo en un navegador web.

## 2.2. Campos editables y no editables

A la hora de programar, muchas veces es necesario incluir en el código web campos que no son editables por el usuario, pero que son informativos para él.

Hay multitud de ejemplos, como:

- El texto plano que indica o informa al usuario.
- El contenido de las tablas de elementos.
- Las listas ya sean numeradas o no numeradas.
- Las imágenes que se muestra en la página.

El hecho de que sean o no editables, simplemente supone una diferencia sustancial entre los permisos de administración y edición, frente a los permisos de un usuario normal, que no siquiera tiene que reconocer el código de dicho sitio web, que para él será totalmente opaco.

Pero esto no es así cuando un programador accede al código de un sitio web, ya que el simple hecho de poder acceder a este, hace que ya tenga todos los permisos para hacer y deshacer a voluntad lo que le parezca.

Sin embargo, a la hora de plantear una plantilla es muy importante que el programador, ya sea él mismo cuando acceda a la plantilla pasado un tiempo, o un programador nuevo que se enfrente a ella, sepa que campos debe editar y cuáles no.

Para ello, hay que diferenciar entre campos editables y campos no editables.

Cómo ya hemos comentado, un programador es, ni más ni menos, que un administrador del código que puede acceder a él sin ningún tipo de restricciones. Entonces, ¿cómo crear campos que los programadores no puedan editar?. La mejor forma de hacerlo es creando comentarios de código que indiquen qué es lo que se puede editar y qué es lo que no.



No hay que olvidar que los comentarios en HTML se crean de forma distinta que en JavaScript y PHP, y que hay que tener cuidado cuando se ponen comentarios para no equivocar la sintaxis.

## UD2

Una vez creada la plantilla, el programador debe detenerse y comentarla, sistemáticamente y de arriba abajo, indicando qué campos son los editables y cuáles son los no editables.

Además de indicar esto, junto a cada campo editable debe indicar necesariamente qué es lo que se espera que deba incluir en ese campo.

Por supuesto, un código con comentarios en los campos no editables sería aún más correcto, y haría que el código superase unos estándares de calidad y elegancia propios de programadores experimentados.

Sin embargo, no es necesario comentar estos últimos elementos.

Es más, es importante tener en cuenta que demasiados comentarios en un código web pueden hacer ilegible el programa cuando se consulte como administrador, así que hay que usar los comentarios con moderación.

Veamos el ejemplo del código de una posible plantilla con campos editables y campos no editables señalados con comentarios. En este caso una plantilla de un formulario que pide los datos personales:

```
<!-- Comienzo de la plantilla -->

<!-- Inicio de datos editables. Colocar tras "action"
la web de destino del formulario -->

<form action="datos_personales.php" method="post">

<!-- Fin de datos editables. -->

<!-- Inicio de datos no editables. -->

<table>

<tr>

<td><label for="nombre">Nombre:</label></td><td><input
type="text" name="nombre" id="nombre" required /> </td>

</tr><tr>

<td><label for="apellidos">Apellidos:</label></td><
td><td><input type="text" name="apellidos" id="apellidos"
required /> </td>

</tr><tr>
```

## UF1304: Elaboración de plantillas y formularios

```

<td><label for="dni">DNI: </label></td><td><input type="text" name="dni" id="dni" required /> </td>

</tr><tr>

<td><label for="domicilio">Domicilio: </label></td><td><input type="text" name="domicilio" id="domicilio" /> </td>

</tr><tr>

<td><label for="poblacion">Poblacion: </label></td><td><input type="text" name="poblacion" id="poblacion" /> </td>

</tr><tr>

<td><label for="cp">Codigo Postal: </label></td><td><input type="text" name="cp" id="cp" /> </td>

</tr><tr>

<td><label for="email">Email: </label></td><td><input type="text" name="email" id="email" /> </td>

</tr><tr>

<td><label for="telefono">Telefono: </label></td><td><input type="text" name="telefono" id="telefono" /> </td>

</tr><tr>

<td></td><td><input type="submit" value="Enviar" /></td>

</tr>

</table>

</form>

<!-- Fin de datos no editables. -->

<!-- Fin de la plantilla -->

```

## UD2

### 2.2.1. Definir y crear los campos susceptibles de cambios en una plantilla

Aunque a estas alturas el alumno debe tenerlo claro, lo primero que debe saber es qué es un campo modifiable.



**Un campo modifiable o susceptible de cambio en una plantilla** es un campo que debe ser modificado por el programador que utiliza la plantilla cuando aplica o añade su código a su propio sitio web.

Una vez que se sabe lo que es un campo modifiable o susceptible de cambio, el programador debe saber cuándo añadirlos a la plantilla. Cuando se crea una plantilla, el programador debe tener claro dos cosas muy importantes a la hora de su aplicación:

- Cuál va a ser la utilidad de la plantilla.
- Qué elementos debe incluir en la plantilla.

Una vez que el programador tiene claro estos dos puntos, es cuando debe empezar a programar el código de la plantilla en sí. Sin embargo, toda plantilla, como ya hemos visto, puede tener campos modificables y campos no modificables. Cuando el programador escriba un campo modifiable debe seguir el siguiente protocolo.

Codificar el campo modifiable tal cual.

Añadir las variables más genéricas e identificativas posibles al campo modifiable.

Aislar el campo modifiable del resto de elementos no modificables.

Poner un comentario de inicio en el campo modifiable para indicar, no sólo su función, sino también lo que debe incluirse en el valor de ese campo.

Poner un comentario final después del campo modifiable, para informar al programador que consulte el código en el futuro donde acaba este.

Una vez seguidos estos pasos, el programador debe seguir con la plantilla, y volver a repetir el protocolo cuando se encuentre con otro campo modificable.



Una plantilla web puede no tener ningún campo modificable, así que estos no son esenciales para crearlas. Pero sí es importante que se añadan campos modificables siempre que sea necesario.

---

Si el código de la plantilla está bien hecho, debería poder identificarse de un simple vistazo cuales son los campos editables, simplemente leyendo los comentarios.

Y lo más importante, si se usa una plantilla con campos susceptibles de cambios en el diseño de un sitio web, además de cambiarlos como es debido, se deben eliminar los comentarios que hagan referencias a la plantilla.

### 2.2.2. Definir y crear los campos no modificables en una plantilla

Al otro lado de los campos susceptibles a ser modificados en una plantilla están los campos no modificables.



**Un campo no modificable o que no debe ser cambiado en una plantilla** es un campo que nunca debe ser modificado por el programador que utiliza la plantilla cuando aplica o añade su código a su propio sitio web.

---

Una vez que se tiene claro lo que es un campo no modificable, el programador debe saber cuándo añadirlos a la plantilla, y sobre todo, como hacerlo de forma que no los tenga que modificar.

## UD2

A igual que cuando se creó una plantilla con campos no modificables, cuando se crea una plantilla, el programador debe tener claro dos cosas muy importantes a la hora de su aplicación:

- Cuál va a ser la utilidad de la plantilla.
- Qué elementos debe incluir en la plantilla.

Esto, como se puede ver, es exactamente igual que cuando se programaban campos modificables, pero es que en realidad ambos campos están muy relacionados entre sí. Ya que ambos campos coexisten en todas las plantillas.

En toda plantilla, como ya debería saber el alumno, puede haber campos modificables y campos no modificables. Pero los campos no modificables son los únicos campos que están siempre presentes en todas las plantillas.



Los campos no modificables, al contrario que los campos sensibles a modificación, están siempre presente en las plantillas.

---

Cuando el programador escriba un campo modificable debe seguir el siguiente protocolo.

Codificar el campo no modificable tal cual.

Identificar, si es que hay, a los campos modificables para que no quenga error de cuáles son estos frente a los no modificables.

Una vez seguidos estos pasos, el programador debe seguir con la plantilla, y tener cuidado de seguir correctamente el protocolo, ya que en ningún caso debe confundirse un campo no modificable con uno sensible a los cambios.

Al igual que ocurre con los campos modificables, si el código de la plantilla está bien hecho, debería poder identificarse de un simple vistazo cuales son los campos no modificables, simplemente leyendo los comentarios.



No hay que olvidar que, aunque normalmente los campos no modificables no tienen comentarios indicando lo que debe hacer el programador con ellos, a veces sí que se incluyen por razones de diseño. Si esto ocurre, cuando se aplica la plantilla al diseño web deben eliminarse estos comentarios.

---

Igualmente, si los comentarios de comienzo y final de la plantilla también deben ser eliminados.

### 2.3. Aplicar plantillas a páginas web

Cuando se tiene un sitio web, aplicar una plantilla tiene diversos grados de complejidad, dependiendo del tipo de plantilla que se quiera incluir.

Si tenemos un sitio web programado en HTML, la plantilla que debe incluirse es simplemente un código anidado en nuestro propio sitio web, y tendrá la complejidad relativa que tenga el elemento a introducir y el conocimiento que tenga el programador de ese elemento.

Por ejemplo, si tenemos esta página con un formulario de ingreso:

```
<html>
<head>
    <title>Ejemplo de formulario de ingreso.</title>
</head>
<body>
    <form action="ingreso.php" method="post">
        Usuario:
        <input type="text" size="30" maxlength="30"
            name="usuario" value="Usuario" />
    </form>
</body>
</html>
```

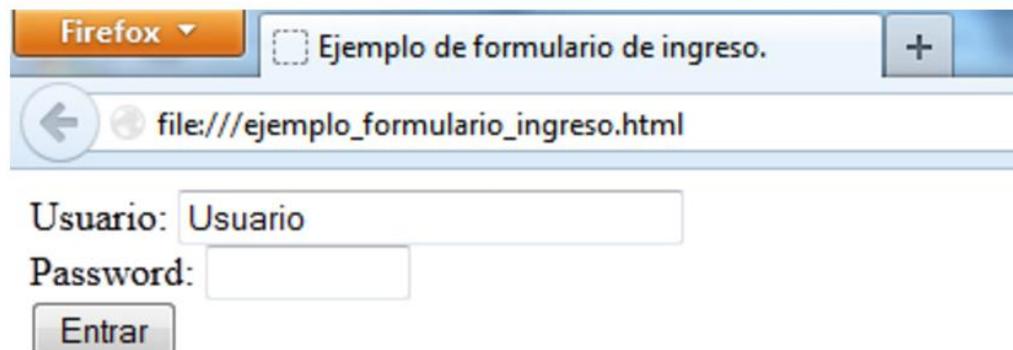
## UD2

```

<br/>
Password:
<input type="password" size="10" maxlength="10"
name="password" />
<br/>
<input type="submit" value="Entrar" />
</form>
</body>
</html>

```

Obtendremos el siguiente resultado cuando la ejecutamos:



Captura de pantalla del resultado que debe dar el código al ejecutarlo en un navegador web.

Pero, ¿y si quisiésemos mostrar un mensaje emergente al cargar la página?

Tendríamos que incluir un código en JavaScript que quizás desconocemos, o incluso aunque conocemos JavaScript y lo hayamos programado anteriormente, se llevaría un buen pedazo del tiempo de trabajo el programarlo.

Sin embargo, si accedemos a una plantilla JavaScript de lo que hay que incluir en un código para poder añadir una plantilla con una ventana emergente, la solución sería mucho más sencilla, tanto en complejidad como en tiempo.

Y resulta que, como buenos programadores que somos, guardamos hace unos meses una plantilla JavaScript con la función que necesitamos en estos momentos:

```
<!-- Comienzo de la plantilla -->

<html>

<head>

    <title>Ejemplo de plantilla JavaScript.</title>

    <script language="JavaScript">

        function mensaje ()
        {

            <!-- Comienzo de la zona editable, cambiar "Mensaje" por el texto que se quiera mostrar -->

            window.alert("Mensaje")

            <!-- Fin de la zona editable -->

        }

    </script>

</head>

<body onload="mensaje()">

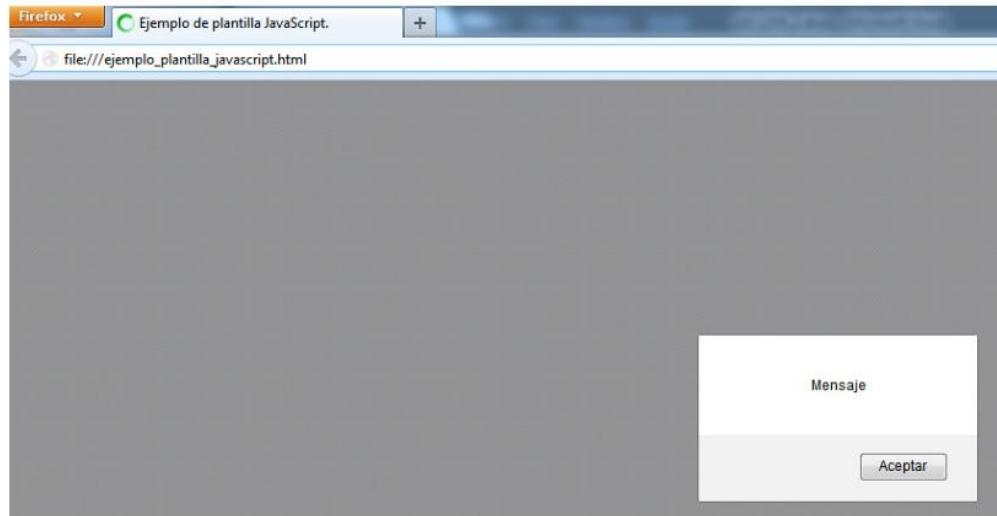
</body>

</html>

<!-- Fin de la plantilla -->
```

Captura de pantalla del resultado que debe dar el código al ejecutarlo en un navegador web.

## UD2



Con esta plantilla ya tenemos el código JavaScript para la función que queremos añadirle a nuestro código web HTML.

Con el código de esta plantilla, ya sólo tendríamos que incluirlo en nuestro programa web original, y cambiar los elementos modificables, que en este caso sólo es uno.

Por ejemplo, si quisiésemos añadir un mensaje emergente que dijese "bienvenidos a nuestra web" el código quedaría así:

```
<html>
<head>
    <title>Ejemplo de formulario de ingreso con mensaje de bienvenida.</title>
    <script language="JavaScript">
        function mensaje ()
        {
            window.alert("Bienvenidos a nuestra web")
        }
    </script>
</head>
<body onload="mensaje()">
```

## UF1304: Elaboración de plantillas y formularios

```

<form action="ingreso.php" method="post">

    Usuario:
    <input type="text" size="30" maxlength="30"
           name="usuario" value="Usuario" />
    <br/>

    Password:
    <input type="password" size="10" maxlength="10"
           name="password" />
    <br/>

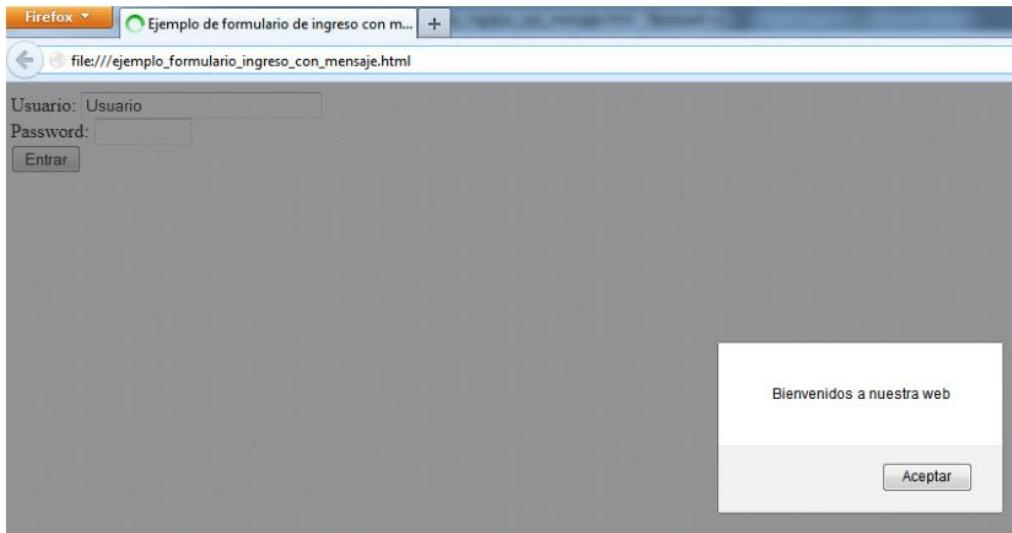
    <input type="submit" value="Entrar" />
</form>

</body>

</html>

```

Captura de pantalla del resultado que debe dar el código al ejecutarlo en un navegador web.



Simplemente añadiendo el código de la plantilla y eliminando los comentarios, hemos podido insertar un mensaje emergente en la web.

Este es el protocolo estándar que se utiliza en la inclusión de plantillas web.

## UD2

Pero esta no es la única forma de incluir plantillas en la programación web hoy en día, debido a la popularidad que están alcanzando ciertos CMS.



Un CMS o Sistema de Gestión de Contenidos (Content Management System en inglés, de ahí vienen las iniciales) es un programa que permite, vía web, crear una estructura fija y visual para la creación y posterior administración de contenidos web, como blogs, foros, tiendas online, servidores de noticias, etc.

Los CMS son, simplemente, programas creados mediante programación web que permiten la creación de páginas web muy fácilmente, y que además son muy vistosas visualmente. Esos programas controlan las bases de datos donde se alojan los sitios web, y además suelen permitir controlar de manera independiente el contenido y el diseño.

Los Sistemas de Gestión de Contenidos más utilizados actualmente en internet son:

- Wordpress.
- Joomla.
- Drupal.
- Prestashop.

Aunque hay muchos otros, como por ejemplo Serendipity, Moodle o Mediawiki.

Cada uno de ellos suele enfocarse en ocupar un hueco en las necesidades tanto de los creadores y administradores de las páginas web, así como de los usuarios.

Por ejemplo, Wordpress es un CMS enfocado a los blogs, aunque hoy en día se pueden encontrar muchos sitios webs creados con Wordpress que son auténticas obras de arte, todo ello gracias, en parte, a las plantillas web.

Joomla es un CMS que está diseñado para crear sitios web interactivos y dinámicos. Pudiendo crear o eliminar contenido mediante un simple panel de control, en lugar de tener que andar modificando el código web.

Drupal es un CMS de carácter libre, por lo que está diseñado para ser modular y poder hacerse con él multitud de herramientas web, como foros, encuestas, blogs o páginas webs normales al uso. Pero es especialmente utilizado para crear y gestionar comunidades en la web.

Prestashop es un CMS especialmente diseñado para crear páginas webs de ventas de productos online. Mediante un panel de control se pueden añadir productos, los métodos de pago, las fotos, e incluso gestionar la compra de dichos productos con sistemas de pago como Paypal.

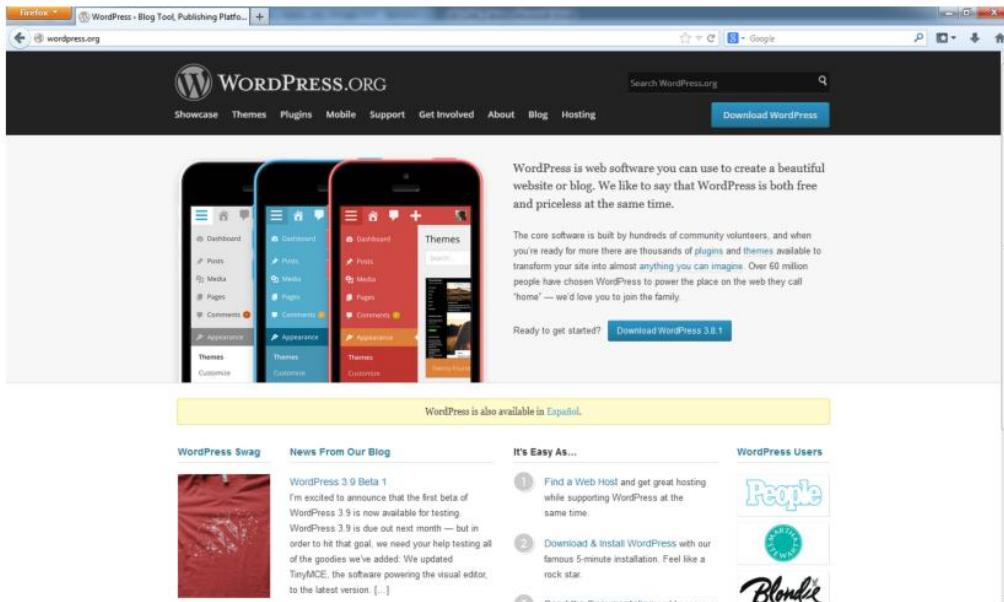
Se podría decir que los CMS, salvando las distancias, actúan en cierto sentido como las plantillas, facilitando el trabajo al programador y administrador web.

Sin embargo, su gran pega es que el aspecto de las páginas creadas mediante CMS son claramente reconocibles debido a su popularidad en la red, ya que existen cientos de miles de páginas creadas con estas herramientas.

Ahí es donde entran en juego las plantillas especialmente creadas para los CMS, que deben ser instaladas en el servidor donde esté la página web.

Wordpress, como se ha explicado anteriormente, es un CMS diseñado y programado específicamente para la creación de blogs.

En su página web se puede bajar la última versión del software que es fácilmente instalable en un servidor web, y que puede ser configurado con facilidad, simplemente asignándole una base de datos con la que pueda trabajar.



## UD2

Captura de pantalla de la página principal de Wordpress realizada con el navegador Mozilla Firefox, donde se pueden ver enlaces para bajarse la última versión de su CMS para instalarlo en un servidor web.

Tras instalar este CMS, el administrador web tiene varias opciones, vía temas del mismo, para modificar el aspecto de la página web usando temas, pero las opciones son muy limitadas.

Lo mejor que puede hacer un programador web cuando diseña una página web con este CMS es instalarle una plantilla adecuada para el diseño que necesita. Para conseguir esta plantilla tiene dos opciones:

- Descargar una plantilla gratuita de internet.
- Comprar una plantilla en internet.

En ambos casos un programador web habrá creado la plantilla de forma independiente, con cada uno de sus elementos y el diseño adecuado, especialmente para que otras personas las puedan usar en sus webs.

Por supuesto, siempre está la opción de que el programador diseñe por sí mismo la plantilla para Wordpress, pero estamos hablando ya de una potente herramienta de diseño que requiere conocimientos, no sólo en varios lenguajes de programación web como son PHP, SQL, HTML o JavaScript, sino también en herramientas especializadas de diseño como son Adobe Photoshop o Gimp.



No hay que olvidar que, aunque Wordpress es un Sistema de Gestión de Contenidos especialmente diseñado para la creación de blogs, en realidad es tan potente que con una buena plantilla puede ser usado hasta para diseñar las páginas webs con apariencia más actual en internet.

---

Para instalar una de estas plantillas descargadas el administrador web tiene que seguir los siguientes pasos:

- En el Panel de Control acceder a la sección de Apariencia.
- Entra en el apartado de Temas.
- Pulsar con el ratón en la pestaña Instalar Temas.

## UF1304: Elaboración de plantillas y formularios

- Se le da a Seleccionar Archivo y se selecciona el Tema o Plantilla que se ha descargado o comprado en la red.
- Se selecciona Instalar ahora.
- En ese momento se podrá seleccionar entre las opciones que salen para previsualizar.



Hay que instalar los temas, siempre que se pueda, en las páginas webs antes de añadirle contenido, porque si no esto podría retocar, e incluso inutilizar el trabajo realizado anteriormente, como veremos más adelante.

Joomla, como se ha explicado anteriormente, es un CMS que está diseñando para crear sitios web interactivos y dinámicos. Pudiendo crear o eliminar contenido mediante un simple panel de control, en lugar de tener que andar modificando el código web.

En su página web se puede bajar la última versión del software que es fácilmente instalable en un servidor web, y que puede ser configurado con facilidad, simplemente asignándole una base de datos con la que pueda trabajar.



## UD2

Captura de pantalla de la página principal de Joomla realizada con el navegador Mozilla Firefox, donde se pueden ver enlaces para comprobar cómo funciona el propio CMS y para descargarlo.

Tras instalar Joomla, al igual que pasaba con Wordpress, el administrador web tiene varias opciones, vía plantillas del mismo, para modificar el aspecto de la página web usando temas, pero las opciones son muy limitadas.

Al igual que pasaba con Wordpress, lo mejor que puede hacer un programador web cuando diseña una página web con este CMS es instalarle una plantilla adecuada para el diseño que necesita.

Para conseguir esta plantilla tiene dos opciones:

- Descargar un tema o plantilla gratuita de la red.
- Comprar un tema o plantilla gratuita de la red.

Al igual que ocurría en el CMS anterior, en los dos casos un programador web habrá creado la plantilla de forma independiente, con cada uno de sus elementos y el diseño adecuado, especialmente para que otras personas las puedan usar en sus webs.

También está la opción de que el programador diseñe por sí mismo la plantilla para Joomla, y aunque esto es más sencillo de hacer que con Wordpress, ya que Joomla es un CMS mucho más cercano al código que el anterior, o al menos no es tan opaco, sigue siendo muy complicado hacerlo, ya que sigue teniendo que dominar varios lenguajes de programación web como PHP, SQL, HTML o JavaScript, y además tendrá que seguir recurriendo a las herramientas especializadas de diseño como son Adobe Photoshop o Gimp.

Para instalar en Joomla una de estas plantillas descargadas el administrador web tiene que seguir los siguientes pasos:

- Una vez descargada la plantilla en el ordenador se debe entrar en administración de la página (en joomla-backend).
- En ese menú se va a Instaladores, a continuación a Plantillas y por último a Sitio.
- Se elige una de las opciones que más se adecuen al diseño y se selecciona el archivo para el paquete a instalar.

- Una vez instalado el paquete se va al menú Sitio, a continuación a Administrar plantillas y en ese lugar se selecciona la plantilla que se ha instalado.
- Cuando se le da a la opción Asignar en la siguiente pantalla aparecen todos los fragmentos de la web a los que se les quiere aplicar la plantilla.
- Se guarda y la plantilla ya está instalada.



Al igual que pasaba en Wordpress, hay que instalar los temas, siempre que se pueda, en las páginas webs antes de añadirle contenido, porque si no esto podría retocar, e incluso inutilizar el trabajo realizado anteriormente, como veremos más adelante.

---

Drupal, como se ha explicado anteriormente, es un CMS de carácter libre, por lo que está diseñado para ser modular y poder hacerse con él multitud de herramientas web, como foros, encuestas, blogs o páginas webs normales al uso. Pero es especialmente utilizado para crear y gestionar comunidades en la web.

Además, Drupal tiene la particularidad de que se divide en tres módulos, llamados las “3 C”, que son el Core (Núcleo), el Contributed (Contribuciones) y el Custom (Personalizados), y en este último módulo donde está el código desarrollado por el creador del sitio.

Drupal puede ser descargado en su página web, donde se encuentra la última versión del software que es fácilmente instalable en un servidor web, y que puede ser configurado con facilidad, simplemente asignándole una base de datos con la que pueda trabajar.

Captura de pantalla de la página principal de Drupal realizada con el navegador Mozilla Firefox, donde se pueden ver las distintas opciones que ofrece este CMS y con una opción en su parte superior para descargarlo.

Drupal al instalarlo trae un tema por defecto, que puede ser cambiado por varios temas que también trae el instalador, en la zona de Administración.

## UD2



Lo mejor que puede hacer un programador web cuando diseña una página web con este CMS es instalarle una plantilla adecuada para el diseño que necesita.

Que recordamos que pueden ser: Portales comunitarios, Foros de discusión, Sitios web corporativos, Aplicaciones de Intranet, Sitios personales o blogs, Aplicaciones de comercio electrónico, Directorio de recursos y Sitios de redes sociales.

Para conseguir una de estas plantillas distintas a las que trae por defecto tiene dos opciones:

- Descargar una plantilla gratuita de internet.
- Comprar una plantilla en internet.

En ambos casos, como hemos visto en los CMS anteriores, un programador web habrá creado la plantilla de forma independiente, con cada uno de sus elementos y el diseño adecuado, especialmente para que otras personas las puedan usar en sus webs.

Para instalar una de estas plantillas descargadas el administrador web tiene que seguir los siguientes pasos:

- Se debe copiar el tema descargado en la subcarpeta "themes", dónde esté instalado el servidor web de Drupal.

## UF1304: Elaboración de plantillas y formularios

- En el Panel de Control acceder a la sección de Administrar.
- Entra en el apartado de Apariencia.
- Allí se debe seleccionar el nuevo tema instalado.



Drupal no es una excepción en este sentido. Hay que instalar los temas, siempre que se pueda, antes de añadirle contenido, porque si no esto podría retocar, e incluso inutilizar el trabajo realizado anteriormente, como veremos más adelante.

---

Prestashop, como ya se ha visto en esta Unidad Didáctica, es un CMS especialmente diseñado para crear páginas webs de ventas de productos online. Mediante un panel de control se pueden añadir productos, los métodos de pago, las fotos, e incluso gestionar la compra de dichos productos con sistemas de pago como Paypal.

En su página web se puede bajar la última versión del software que es fácilmente instalable en un servidor web, y que puede ser configurado con facilidad, simplemente asignándole una base de datos con la que pueda trabajar.



## UD2

Captura de pantalla de la página principal de Prestashop realizada con el navegador Mozilla Firefox, donde se pueden ver un enlace para bajarse el CMS y un slider con información sobre la implementación del mismo.

Tras instalar este CMS, el administrador web tiene varias opciones, vía temas del mismo, para modificar el aspecto de la tienda online usando temas, pero las opciones son muy limitadas.

Importante: No hay que olvidar que este Sistema de Gestión de Contenidos está especialmente diseñado para crear sitios webs de tiendas online, lo que supone una gran diferencia con los CMS anteriores, que con la debida implementación podían ser usados para crear cualquier tipo de página web.

Lo mejor que puede hacer un programador web cuando diseña una página web con Prestashop, al igual que ocurre con los CMS anteriores, a pesar de sus diferencias, es instalarle una plantilla adecuada para el diseño que necesita.

Para conseguir esta plantilla tiene las dos opciones que ya conoce el alumno:

- Descargar una plantilla gratuita de internet.
- Comprar una plantilla en internet.

En ambos casos un programador web habrá creado la plantilla de forma independiente, con cada uno de sus elementos y el diseño adecuado, especialmente para que otras personas las puedan usar en sus webs.

Para instalar una de estas plantillas descargadas el administrador web tiene que seguir los siguientes pasos:

- Se descomprime el archivo de la plantilla comprada o descargada.
- Se debe localizar la plantilla en los archivos descomprimidos, que normalmente están instalados en la carpeta “themes/nombre de la plantilla”, pero a veces la plantilla está directamente en una carpeta con el “nombre de la plantilla”.
- Se mueve la carpeta que contiene la plantilla a la carpeta “lugar donde está el servidor/themes”.
- En el Back Office de prestashop hay que activar en “Preferencias”, a continuación “Apariencia” y por último en “Aplicar Nombre del Tema”.
- Se verifica en “backoffice”, y a continuación “modules” que todos los módulos nuevos están instalados.



Algunas plantillas de Prestashop traen una carpeta para reemplazar los módulos que trae prestashop por defecto sobrescribiéndolos con los que trae la carpeta.

---

### 2.3.1. Las plantillas en la web

Las plantillas, además de ser creadas por el mismo programador web, pueden ser descargadas de internet.

Estas plantillas pueden ser de dos tipos:

- Plantillas subidas voluntariamente por diseñadores web que se pueden descargar de forma gratuita.
- Plantillas subidas en plataformas especializadas para vender plantillas que han subido diseñadores web para que las use quien pague por ellas.

Existe una regla respecto a la variabilidad de las plantillas, y es que, mientras más de bajo nivel sea el sistema que el programador web utilice para realizar su página web, más posibilidades de encontrar plantillas gratuitas tendrá.

Por ejemplo, es mucho más sencillo encontrar plantillas gratuitas para HTML y JavaScript, ya que son lenguajes que necesitan ser codificados por un programador con conocimientos medio-altos de la materia.

Sin embargo, encontrar plantillas de CMS, como por ejemplo Wordpress o Prestashop, es mucho más complicado.

Aunque también es cierto que es bastante más sencillo realizar webs visualmente más atractivas usando los CMS más conocidos, ya que gran parte de la codificación más compleja la traen resuelta por defecto, y además muchas de las plantillas con mejor aspecto han sido diseñadas no solo por programadores web expertos, sino también por diseñadores artísticos muy capaces y con gran talento en esa materia.

## UD2

En todo caso siempre que se use una plantilla ajena hay que tener en cuenta varias cosas:

- Realizar esa plantilla le ha supuesto un trabajo, más o menos extenso, a un programador web, así que es importante reconocerle su trabajo. Para hacer eso a veces simplemente hay que poner un enlace a pie de web para enlazar la descarga de la plantilla. Otras veces con un comentario en el código es suficiente.
- Nunca, bajo ninguna circunstancia, se debe descargar una plantilla pirateada, ya que realmente un programador web se estará tirando piedras sobre su propio tejado, ya que nunca se sabe si algún día va a diseñar una él mismo.
- Antes de instalar una plantilla web hay que leerse detenidamente todas sus especificaciones, que normalmente estarán en la propia web de donde se descarga, o en un documento adjunto con la plantilla.
- Hay que tener cuidado con la aplicación de las plantillas en nuestra página web, ya que a veces no son lo que se espera, e incluso si se aplica una plantilla a un CMS a veces es difícil revertir el proceso y volver al estado anterior.

Importante: Antes de realizar un cambio en una página web aplicando una plantilla encontrada en la red siempre se debe hacer una copia de seguridad del trabajo. Esto es quizás la regla más importante de la aplicación de plantillas, ya que no guardar una copia de seguridad del trabajo, aplicar una plantilla, y que luego el proceso no sea reversible, podría haber machacado el trabajo de muchos días de programación.

Por ejemplo, aquí tenemos una plantilla de formularios en HTML bajada de la web:

```
name="htmlform"    method="post"    action="html_form_send.php">

<table width="450px">

</tr>

<tr>

<td valign="top"> <label for="first_name">First Name * </label> </td>
```

## UF1304: Elaboración de plantillas y formularios

```
<td valign="top"> <input type="text" name="first_name" maxlength="50" size="30"> </td>

</tr>

<tr>

    <td valign="top"> <label for="last_name">Last Name * </label> </td>

    <td valign="top"> <input type="text" name="last_name" maxlength="50" size="30"> </td>

</tr>

<tr>

    <td valign="top"> <label for="email">Email Address * </label> </td>

    <td valign="top"> <input type="text" name="email" maxlength="80" size="30"> </td>

</tr>

<tr>

    <td valign="top"> <label for="telephone">Telephone Number </label> </td>

    <td valign="top"> <input type="text" name="telephone" maxlength="30" size="30"> </td>

</tr>

<tr>

    <td valign="top"> <label for="comments">Comments * </label> </td>

    <td valign="top"> <textarea name="comments" maxlength="1000" cols="25" rows="6"></textarea> </td>

</tr>

<tr>

    <td colspan="2" style="text-align:center">
```

## UD2

```

<!-- We are grateful to you for keeping this link in place. thank you. -->

<input type="submit" value="Submit"> ( <a href="http://www.freecontactform.com/html_form.php">HTML Form</a> )

</td>
</tr>
</table>
</form>

```

Véase que en este caso no hay comentarios indicando de dónde se ha bajado el formulario, sin embargo uno de los campos modificables, el que tiene el botón de submit, indica de dónde se ha bajado el mismo.

### 2.3.2. Búsqueda de plantillas en la red

La búsqueda de plantillas en la red puede ser desde muy rápida hasta el nivel de complejidad que requiera el trabajo que busque el programador.

En todo caso, la herramienta principal inicial de estas búsquedas, como ocurre con casi todo en la red hoy en día, se hará con los buscadores web como Google, Bing o DuckDuckGo.

Estas búsquedas se harán especificando lo siguiente:

- Lenguaje de programación en el que se quiere la plantilla.
- Función que quiere que realice la plantilla.
- Si se quiere que actúe en una parte concreta del código o de la web también se debe especificar en la búsqueda.

Por ejemplo, si se quiere encontrar una plantilla para poner un mensaje emergente de alerta en nuestra web utilizando JavaScript, haríamos lo siguiente:

Captura de pantalla de una búsqueda en Google realizada con el navegador Mozilla Firefox, donde se busca "javascript mensaje emergente de alerta".

## UF1304: Elaboración de plantillas y formularios

Aproximadamente 41.000 resultados (0,16 segundos)

Las cookies nos permiten ofrecer nuestros servicios. Al utilizar nuestros servicios, aceptas el uso que hacemos de las cookies.

[Aceptar](#) [Más información](#)

[Aplicación de comportamientos JavaScript incorporados - Ad...](#)  
help.adobe.com/.../WSc78c5058ca073340dcda9110b1f693f21-7b09a.htm... ▾  
Ir a [Aplicación del comportamiento Mensaje emergente](#) - El comportamiento Mensaje emergente muestra una alerta de JavaScript con el mensaje que ...

[Ejemplos prácticos de Javascript \(I\) | Observatorio Tecnológico](#)  
recursostic.educacion.es/observatorio/web/ca/.../490-lorena-arranz ▾  
Visualizar por pantalla un mensaje de bienvenida a la página Web. ... La función confirm visualiza una ventana emergente de confirmación (con el texto que le ... 2 || event.button == 3) y entonces lo que genera es visualizar una alerta.

[Cómo publicar un mensaje emergente en tu sitio web | eHo...](#)  
www.ehowenEspañol.com › Computación y electrónica ▾  
... probable que hayas encontrado una página con un cuadro de alerta emergente. ...  
aprender cómo crear un cuadro de mensaje emergente con JavaScript que .

Entre los resultados obtenidos se va navegando hasta dar con la plantilla que deseamos obtener para incluir en nuestro código web.

En este caso, en la segunda entrada que nos ha otorgado Google nos encontramos con el siguiente código:

[2. Ejemplos](#)

Visualizar por pantalla un mensaje de bienvenida

```
<HTML>
<HEAD>
<TITLE>Ejemplo01.htm</TITLE>

<SCRIPT LANGUAGE="JavaScript">
//Visualizar un mensaje de bienvenida
alert("¡Bienvenido a nuestra página!");
</SCRIPT>

</HEAD>
<BODY>
<a href="Ejemplo02.html">Ir al siguiente ejemplo...</a>
</BODY>
</HTML>
```

Ejemplo 1 . Visualizar por pantalla un mensaje de bienvenida a la página Web.

En nuestro primer ejemplo, simplemente, crearemos un Javascript que lo que hará es visualizar un mensaje de bienvenida cuando el usuario entre a la página Web. Cada vez más, las páginas Webs muestran un mensaje de este tipo para dar la bienvenida cuando se entra en ellas.

Como se puede apreciar el código Javascript se coloca (o se embellece) dentro de uno de HTML, encerrándolo entre las etiquetas <script></script>.

Abre el bloc de Notas, copia el código anterior y guárdalo como Ejemplo01.html. (Si no te apetece hacer esto puedes abrir el mismo desde el directorio Ejemplos I). Después, ejecútalo pulsando doble clic con el ratón.

[Botón para volver hacia atrás](#)

```
<html>
<TITLE>Ejemplo02.htm</TITLE>
<input type="button" value="Atrás" onclick="history.go(-1)">
</html>
```

## UD2

Captura de pantalla del Instituto Nacional de Tecnologías Educativas y Formación del Profesorado, que está en el Ministerio de Educación, Cultura y Deporte, con el navegador Mozilla Firefox.

En este caso ya podemos ver una plantilla de la programación de una ventana emergente en JavaScript que se ejecuta al cargar la página web. Si lo que se quisiese es una plantilla de una ventana emergente al pulsar un botón o realizar un evento especial, habría, o programarlo uno mismo, o buscar mejor otra plantilla que realice tal efecto.

En todo caso, en la red hay multitud de información que pueden ayudar al programador en cualquier tipo de lenguaje. Lo realmente complicado es discriminar la información que no es necesaria y encontrar las plantillas web adecuadas a la información que se está buscando. Para ello, se debe intentar:

- Ser lo más específico posible en la búsqueda de la plantilla que se está buscando.
- Usar las herramientas que tienen los navegadores para buscar cadenas concretas de datos.
- Usar la opción de los navegadores para buscar en sitios concretos de dónde ya se han obtenido plantillas anteriormente y el programador sospecha que pueda encontrar información.

La búsqueda de plantillas es, si cabe, más complicada cuando se buscan para los distintos CMS, si es que se quieren usar los buscadores.

Los propios Sistemas de Gestión de Contenidos suelen tener en sus páginas web o incluso entre las opciones del propio CMS, opciones para comprar plantillas de pago, o aplicar plantillas gratuitas por defecto.

Sin embargo, hoy en día proliferan unas páginas web especialistas en ofrecer a los propios desarrolladores de plantillas una plataforma donde ofrecerlas, mediante pago, al diseñador web que las necesite.

Estas webs actúan como un escaparate donde que diseñan las plantillas ofrecen capturas de su trabajo, descripción del mismo, y hasta simulaciones de sus webs online para que el programador pueda ver cuál es el resultado final.

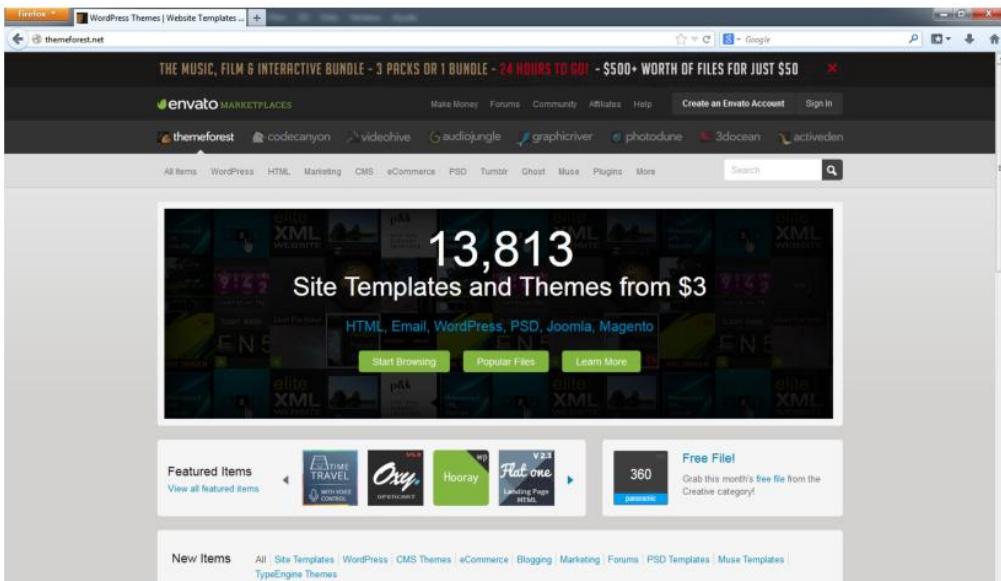
Por ejemplo, en la siguiente web pueden encontrarse plantillas de todos los CMS más conocidos, donde podrán comprarse y descargarse:

## UF1304: Elaboración de plantillas y formularios



Captura de pantalla de la página principal de Template Monster en el navegador Mozilla Firefox.

Y en esta conocida web también pueden encontrarse prácticamente todas las plantillas más utilizadas en la red en las web con mejor diseño en la actualidad:



Captura de pantalla de la página principal de Theme Forest en el navegador Mozilla Firefox.

## UD2

Pero hay decenas de webs parecidas a estas, donde poder encontrar las plantillas a la venta puestas por los propios diseñadores.

En todo caso, estas webs tienen siempre dos elementos en común:

- Una barra de menús en su parte superior donde seleccionar el Sistema de Gestión de Contenidos en el que se quiere encontrar la plantilla.
- Un buscador donde especificar qué tipo de plantilla se quiere encontrar junto con opciones para ajustar la búsqueda al CMS en concreto.

En todo caso, una vez encontrada la plantilla deseada, nada más que hay que pagar, descargarla y proceder a su instalación en el servidor web.



Si el alumno aún no se ha dado cuenta o no es consciente de ello, la mayoría de las mejores herramientas para encontrar plantillas o el desarrollo de diseño web se encuentran en inglés, por ello es muy importante un alto conocimiento de este lenguaje para el desarrollo web.

Es más, muchas veces la búsqueda de una plantilla en castellano en los motores de búsqueda resulta infructuosa, y sin embargo si esa misma búsqueda se realiza en inglés, se obtienen muchos más resultados o incluso se pueden encontrar soluciones que no se encuentran en español, así que el uso del inglés debería ser algo vital para cualquier programador web.

### 2.3.3. Adaptación de plantillas a páginas web

Muchas veces lo más complicado de una plantilla no es encontrarla, si no aplicarla a la propia página web, ya que el código es demasiado denso, o no encaja de forma sencilla en el creado por el propio programador web, ya sea porque hay que modificar elementos ya de por sí muy recargados, o simplemente porque recoge supuestos que no han sido planteados de forma inicial por el propio programador.

En todos estos casos es muy importante que el programador tenga:

## UF1304: Elaboración de plantillas y formularios

- Un dominio bastante alto de los lenguajes de la plantilla y de los elementos a los que quiere aplicar dicha plantilla.
- Flexibilidad para poder adaptar su código y amoldar el de la plantilla a su propio sitio web.
- Capacidad de discriminación de opciones, y no quedarse siempre con la primera opción que encuentre sin ver más alternativas, ya que no siempre la primera plantilla o el primer código que se encuentra es la mejor solución.

En todo caso, una vez descargada la plantilla el programador debe identificar en qué lugar de su código debe situarla, y aplicarla. Por ejemplo, el código JavaScript encontrado en la búsqueda anterior debería meterse en la cabecera de la página, como en este caso:

```

<html>

  <head>

    <title>Ejemplo de formulario simple.</title>

    <SCRIPT LANGUAGE="JavaScript">

      //Visualizar un mensaje de bienvenida

      alert("¡Bienvenido a nuestra página!");

    </SCRIPT>

  </head>

  <body>

    <form action="http://www.paginadeejemplo.com/formulario.php" method="post">

      Nombre: <input type="text" name="nombre" value="" />
      <br/>

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

    </form>

  </body>

</html>
```

## UD2



Esta no tenía que haber sido la mejor solución o la primera que encontrásemos en la web, sin embargo es la que hemos adaptado a nuestro código.

---

A la hora de aplicar plantillas a los CMS hay que ser mucho más cuidadoso.

Por ejemplo, una página web creada con Wordpress, que como ya debería saber el alumno, es un CMS diseñado y programado específicamente para la creación de blogs, pero que en realidad puede ser usado para multitud de aplicaciones e incluso para crear páginas web con apariencia profesional con mucha facilidad, puede llegar a ser muy peligroso para el contenido web creado anteriormente.

Por ejemplo, si se ha creado una página web con aspecto de blog mediante Wordpress, y se le aplica una plantilla para cambiar su aspecto a una página web con slider principal y distintas ventanas para mostrar la información, de repente podría ocurrir que gran parte del trabajo realizado desapareciese.

Es más, podría darse el caso que en la nueva plantilla la información a mostrar se introdujese vía "widgets", por ejemplo, en lugar de vía "entradas", lo que podría significar que el trabajo de varias semanas de programación tendría que ser trasladado a otra plataforma.

Por ello, especialmente con este Sistema de Gestión de Contenidos, el programador debe ser muy cuidadoso a la hora de aplicar las plantillas. Aquí tenemos un ejemplo de lo que pueda cambiar una página web con Wordpress cuando se le aplica una plantilla:

UF1304: Elaboración de plantillas y formularios



Captura de pantalla de la demostración de una plantilla realizada por un diseñador web y puesta a la venta en Template Monster el navegador Mozilla Firefox.

Por supuesto, el programador debe ser muy cuidadoso también a la hora de eliminar una plantilla web de su diseño, ya que podría perder mucha información en el proceso.

Por ejemplo, si el diseñador instalara en Wordpress una plantilla de venta de productos, estuviese durante muchos días codificando los productos para vender, y a continuación borrase esa plantilla, perdería todo el trabajo realizado al respecto.



Hay que remarcar que es muy importante guardar copias de seguridad en cualquier diseño web antes de realizar un cambio en el mismo que pueda acarrear la pérdida de la información que ha sido codificada por el programador.

---

La instalación de una plantilla, en una página web ya avanzada, es similar a la de la aplicación de una plantilla a una web recién creada:

## UD2

En el Panel de Control acceder a la sección de Apariencia.

- Entra en el apartado de Temas.
- Pulsar con el ratón en la pestaña Instalar Temas.
- Se le da a Seleccionar Archivo y se selecciona el Tema o Plantilla que se ha descargado o comprado en la red.
- Se selecciona Instalar ahora.
- En ese momento se podrá seleccionar entre las opciones que salen para previsualizar.

En cualquier caso se debe comprobar siempre el resultado antes de seguir avanzando en el diseño del sitio web.

En las webs creadas con Joomla, al igual que pasa con las creadas con Wordpress, hay que ser muy cuidadoso a la hora de aplicar una plantilla. Esto ocurre porque, como ya sabe el alumno, Joomla es un CMS que está diseñado para crear sitios web interactivos y dinámicos. Pudiendo crear o eliminar contenido mediante un simple panel de control, en lugar de tener que andar modificando el código web.

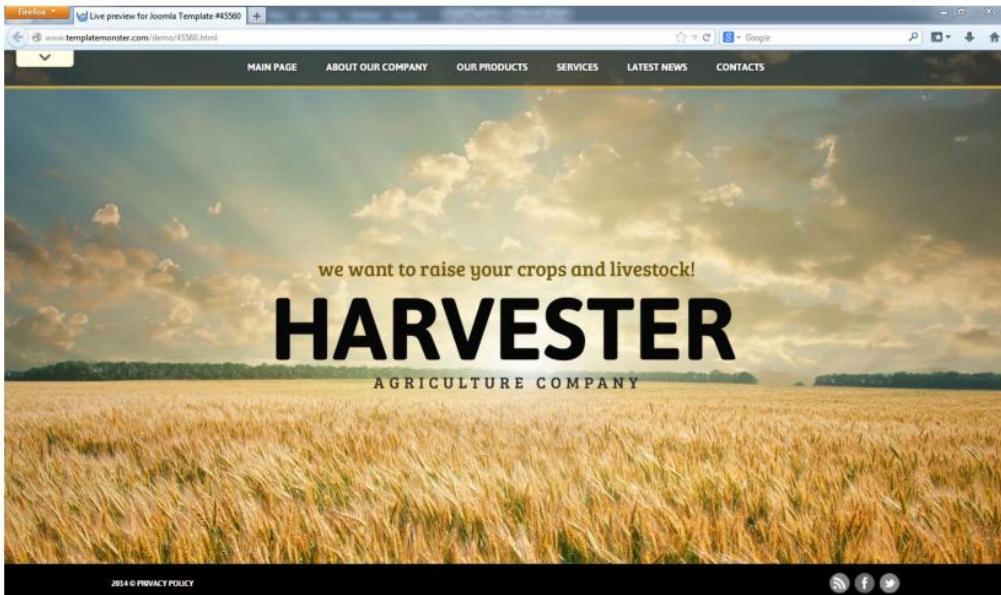
Esto quiere decir que el cambio de apariencia drástico aplicado mediante una plantilla puede significar la pérdida de mucho trabajo de diseño y dinamismo a la página web.

Por ejemplo, pongamos el caso que el programador diseña una página web al estilo clásico, con un menú a la izquierda con multitud de opciones y de subopciones. Varias aplicaciones a la derecha para mostrar, por ejemplo, enlaces, la fecha, la hora, etc. Y luego en medio contenido, que se basa sobre todo en el texto.

De repente, el programador quiere cambiar el diseño, ya sea por opinión propia, o por una petición del cliente, que quiere una web muy parecida a uno de sus competidores, que le ha gustado.

Entramos en una página web de plantillas para Joomla, y navegando entre todas decidimos comprar la siguiente:

UF1304: Elaboración de plantillas y formularios



Captura de pantalla de la demostración de una plantilla realizada por un diseñador web y puesta a la venta en Template Monster el navegador Mozilla Firefox.

Una vez instalada el programador se da cuenta de dos cosas:

- Todos los elementos creados hasta ese momento no le sirven para nada.
- Todo el código introducido debe cambiarse de lugar y de aspecto para que concuerde con la nueva plantilla.

Como es obvio, este resultado es un desastre para cualquier programador web, ya que ha malgastado sin obtener ningún resultado muchas horas de su trabajo.



Cuando se acepte el desarrollo de una página web de un cliente deben dejarse lo más claro posible todos los puntos del diseño final de la misma, para evitar que ocurran situaciones como esta y todo el trabajo de varios días no sirva para nada.

## UD2

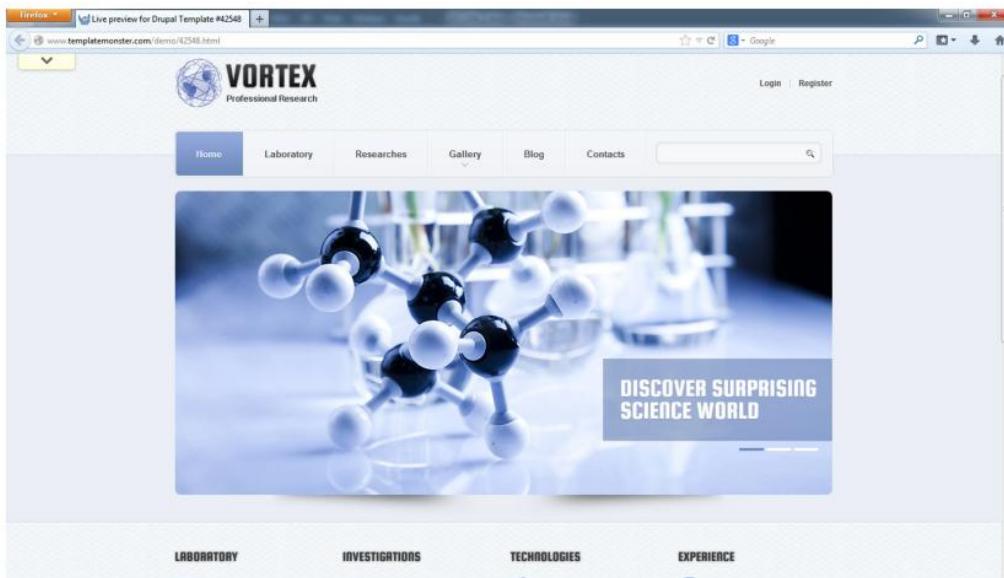
Al igual que ocurría cuando se instalaba una plantilla en Joomla con una página recién creada, para instalar una plantilla en una página ya avanzada se deben seguir los siguientes pasos:

- Una vez descargada la plantilla en el ordenador se debe entrar en administración de la página (en joomla-backend).
- En ese menú se va a Instaladores, a continuación a Plantillas y por último a Sitio.
- Se elige una de las opciones que más se adecuen al diseño y se selecciona el archivo para el paquete a instalar.
- Una vez instalado el paquete se va al menú Sitio, a continuación a Administrar plantillas y en ese lugar se selecciona la plantilla que se ha instalado.
- Cuando se le da a la opción Asignar en la siguiente pantalla aparecen todos los fragmentos de la web a los que se les quiere aplicar la plantilla.
- Se guarda y la plantilla ya está instalada.

Con Drupal existe exactamente el mismo problema que con los Sistemas de Gestión de Contenidos anteriores. Ya que, al ser un CMS de carácter libre, diseñado para ser modular y poder hacerse con él multitud de herramientas web, como foros, encuestas, blogs o páginas webs normales al uso, y que además está especialmente utilizado para crear y gestionar comunidades en la web. La cantidad de opciones que ofrece es tan tremenda, que el hecho de cambiar de una a otra puede dar un cambio de 180 grados al diseño.

Por ejemplo, si el diseño inicial del programador web fuese la de crear una comunidad web basada en la interacción social mediante un foro, pero de repente este se diese cuenta que la página necesita un diseño distinto y buscase una plantilla para darle un aspecto más de una web moderna al uso, como por ejemplo esta opción:

## UF1304: Elaboración de plantillas y formularios



Captura de pantalla de la demostración de una plantilla realizada por un diseñador web y puesta a la venta en Template Monster el navegador Mozilla Firefox.

No sólo podría encontrarse que el trabajo que ha realizado es prácticamente desecharable, si no es que es seguro 100% que lo será, ya que al ser un CMS con tantas opciones, aunque se esté bajo el mismo paraguas, en teoría, de diseño, el resultado final es tan distinto que prácticamente es como si se hubiese usado otro CMS para programarlo.

Importante: En los Sistemas de Gestión de Contenidos como Drupal es mucho más importante, si cabe, saber exactamente qué es lo que se quiere obtener como resultado final. Y esto es así debido a que la alta capacidad de creación de multitud de resultados web también es un arma de doble filo, ya que puede parecer, a priori, que la forma de obtener el resultado es similar, pero no hay nada más lejos de la realidad, ya que la forma de programar un resultado u otro puede ser sumamente distinta.

En todo caso, el dominar la programación y la búsqueda de plantillas en la web para Drupal, la alta cantidad de opciones y las distintas alternativas de diseño hacen que aprender a usar este CMS sea mucho más útil que aprender otros, por lo que es una gran opción, unido a que es un programa libre bajo la licencia GNU/GPL.

## UD2



[La Licencia Pública General de GNU](#) garantiza a los usuarios la libertad para usar, estudiar, copiar y modificar el software que está inscrito a esta licencia, y por ello es la más ampliamente usada en el mundo del software.

---

Para instalar una de estas plantillas descargadas el administrador web tiene que seguir los mismos pasos que en la instalación de las plantillas en páginas web recién creadas:

- Se debe copiar el tema descargado en la subcarpeta “themes”, dónde esté instalado el servidor web de Drupal.
- En el Panel de Control acceder a la sección de Administrar.
- Entra en el apartado de Apariencia.
- Allí se debe seleccionar el nuevo tema instalado.



El logo de Drupal, que es una gota con una carita en ella, fue concebido inicialmente como una simple gota dentro de un círculo, pero que finalmente se optó por ponerle una cara a una gota para darle un toque más simpático al logo.

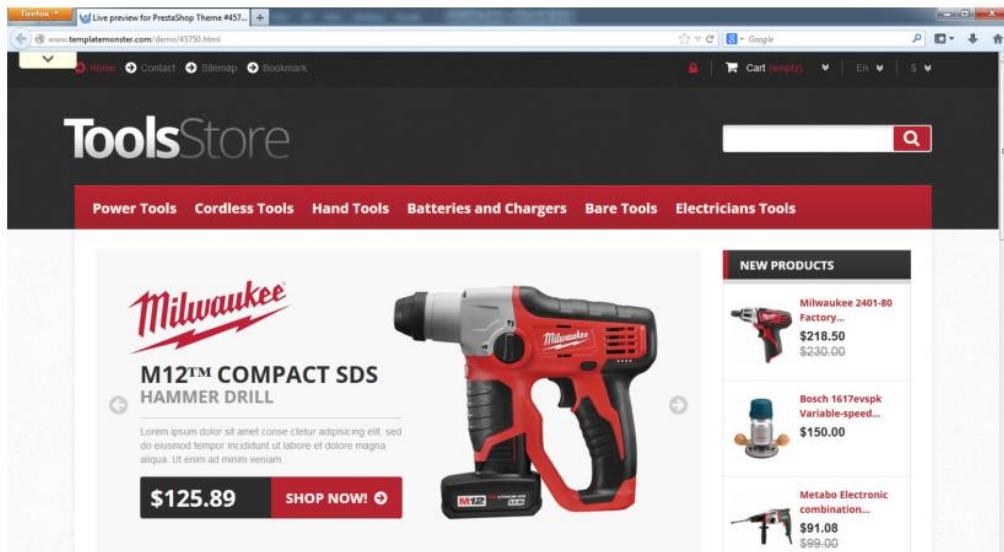
---

Y por último, y para ver que no todos los Sistemas de Gestión de Contenidos son iguales, nos encontramos con la aplicación de una plantilla a una web ya creada y con mucha programación en Prestashop.

Este CMS está especialmente diseñado para crear páginas webs de ventas de productos online. Mediante un panel de control se pueden añadir productos, los métodos de pago, las fotos, e incluso gestionar la compra de dichos productos con sistemas de pago como Paypal. Y, al contrario que el resto de CMS vistos hasta ahora, no es recomendable usarlo para crear páginas web más allá de su función original.

## UF1304: Elaboración de plantillas y formularios

Por lo tanto, precisamente debido a su poca flexibilidad, el hecho de aplicar una plantilla sobre una página web ya creada va a traer problemas menores de diseño. Por ejemplo, si el programador web ha creado una tienda online con el tema por defecto que tiene Prestashop, y decide instalar esta plantilla:



Captura de pantalla de la demostración de una plantilla realizada por un diseñador web y puesta a la venta en Template Monster el navegador Mozilla Firefox.

Todos los productos introducidos anteriormente en la web se mantendrán tal cual, sin necesidad de realizar ningún cambio en ellos. Es más, todos los métodos de pago programados, los productos destacados, las ventas realizadas a los clientes, las compras a proveedores, los datos personales de la empresa, las opciones de administración, los addons, etc, se mantendrán tal cual estaban antes de que el administrador web aplicase dicha plantilla.



Algunos objetos de diseño como el slider, las imágenes de la presentación, la cabecera, etc. sí que tendrán cambios que deberán ser revisados por el programador web. Pero en todo caso son cambios menores comparados con los que se producían en otros Sistemas de Gestión de Contenidos al aplicar las plantillas.

## UD2

Por lo tanto, a la hora de aplicar una plantilla web, en este CMS no es tan relevante hacerlo al inicio de su programación. Aunque, como siempre, es recomendable hacerlo cuanto antes.

Finalmente, para instalar una de estas plantillas descargadas el administrador web tiene que seguir los mismos pasos que realizó para instalar la plantilla en una página web recién creada con Prestashop:

- Se descomprime el archivo de la plantilla comprada o descargada.
- Se debe localizar la plantilla en los archivos descomprimidos, que normalmente están instalados en la carpeta "themes/nombre de la plantilla", pero a veces la plantilla está directamente en una carpeta con el "nombre de la plantilla".
- Se mueve la carpeta que contiene la plantilla a la carpeta "lugar donde está el servidor/themes".
- En el Back Office de prestashop hay que activar en "Preferencias", a continuación "Apariencia" y por último en "Aplicar Nombre del Tema".
- Se verifica en "backoffice", y a continuación "modules" que todos los módulos nuevos están instalados.

Área: informática y comunicaciones

# UD2

## Lo más importante

Tras finalizar esta Unidad Didáctica, el alumno debe ser capaz de dominar la creación, búsqueda y aplicación de plantillas web.

- En concreto, el alumno debe ser capaz de describir todas las funciones y las características principales de una plantilla web.
- Así mismo, el alumno debe poder saber de memoria la descripción de lo que es una plantilla web, que viene ligada a su utilidad y a su aplicación en diseños web realizados por uno mismo o por terceros.
- Los elementos de una plantilla web deben quedar muy claros, en especial como aplicarlos respetando las etiquetas, como introducir los comentarios destinados a identificarlos, y sobre todo a diferenciar los campos modificables y los campos no modificables.
- Una vez conocidos estos elementos, el alumno debe poder describir y definir la estructura de dichos elementos de las plantillas, así como su organización en los distintos programas realizados en código web.
- Todo esto servirá para que el alumno pueda identificar, al ver a simple vista una plantilla, cuáles son sus partes modificables y cuales las que no.
- Para, finalmente, poder utilizarlas y aplicarlas en los propios diseños, y crearlas uno mismo con la intención de crear un repositorio de plantillas que puede servir para ahorrar tiempo y trabajo al programador web en el futuro.

## UF1304: Elaboración de plantillas y formularios

- Tras hacer esto el alumno debe introducirse y dominar las diferencias entre los dos principales elementos de las plantillas, como son los campos susceptibles de cambios o también llamados campos editables o modificables, y los campos no modificables.
- Los campos susceptibles de cambios deben poder ser identificados en una plantilla, y para ello se utilizarán especialmente herramientas dispuestas para ello. En las plantillas de programación web estándar principalmente se usarán para ello los comentarios.
- Los campos no modificables serán totalmente distintos a los anteriores, y se identificarán, además de por el hecho de que no se podrán modificar en plantillas específicas creadas para ello, en que no tienen comentarios cuando se localizan en plantillas en lenguajes de programación web de más bajo nivel.
- Por último el alumno ha aprendido como aplicar las plantillas a las páginas web, tanto en páginas realizadas con lenguajes de programación web estándar, como pueden ser HTML, JavaScript o PHP, como en páginas realizadas con Sistemas de Gestión de Contenido, llamados CMS, especialmente en los más conocidos y utilizados en la red.
- El alumno habrá aprendido a identificar las plantillas de cada uno de estos lenguajes web y CMS, y además tendrá la capacidad suficiente para buscar la plantilla adecuada en cada uno de estos entornos, ya sea mediante la utilización de buscadores web como Google, Bing o DuckDuckGo, o en las páginas especializadas para la compra de plantillas web donde los propios diseñadores de las mismas las ponen a disposición de los creadores web para su instalación.
- Por último, el alumno deberá saber cómo buscar las plantillas más adecuadas para el problema que intenta solucionar en su propio diseño web, descargarse esas plantillas de la plataforma adecuada, ser capaz de identificar en qué lugar de su página web debe instalarla, y como seguir los protocolos finales para dejar la plantilla instalada sin que esto interfiera en el código ya creado, o al menos lo haga en la menor medida posible.

# UD2

## Autoevaluación

1. ¿Para qué sirven las aplicaciones destinadas para la creación, construcción, diseño y edición de páginas web?
  - a. Para facilitar el trabajo al programador y que pueda observar el proceso de su creación de forma visual.
  - b. Para eliminar el trabajo del programador y que no tenga que conocer ninguna línea de código del lenguaje web.
  - c. Para que el programador web pueda trabajar sin el uso de plantillas web y obtener páginas web del mismo aspecto visual.
  - d. Estas aplicaciones están obsoletas y no tienen uso en la actualidad.
2. ¿Para qué están diseñadas las plantillas?
  - a. Para hacer código más complicado del que puede hacer el programador.
  - b. Para introducir mucho código de programación web en menos espacio.
  - c. Para poder hacer las mismas aplicaciones web en lenguajes de programación distintos.
  - d. Para ahorrarle tiempo al programador.

3. ¿Qué tienen en común los elementos modificables en una plantilla?

- a. Todos deben empezar con la misma inicial para poder ser identificados.
- b. Deben ser nombrados de forma que se sepa claramente qué función cumplen en la plantilla.
- c. Deben estar programados en el mismo lenguaje de programación.
- d. Tienen una extensión común específica.

4. Por qué hay plantillas que no deben modificarse?

- a. Porque el código es fijo.
- b. Porque el código no es fijo.
- c. Porque el código no puede ser modificado en una plantilla.
- d. Porque el código es abstracto.

5. ¿Cuál es la gran ventaja de la utilización de una plantilla básica al crear una nueva página de HTML?

- a. No necesita demasiados comentarios para indicarles a futuros programadores qué es cada cosa.
- b. El programador podrá usar la misma estructura de la plantilla básica para crear la página web.
- c. Podrá incluir el lenguaje con otros lenguajes web como JavaScript o PHP con más facilidad.
- d. No tiene ventajas.

6. ¿Cómo se crean los comentarios en HTML?

- a. Igual que en JavaScript.
- b. Igual que en PHP.
- c. Igual que en JavaScript y PHP.
- d. De forma distinta que en JavaScript y PHP.

## UD2

7. ¿Qué es un campo modifiable o susceptible de cambio en una plantilla?
  - a. Un campo que nunca debe ser modificado por el programador que utiliza la plantilla cuando aplica o añade su código a su propio sitio web.
  - b. Un campo que debe ser modificado por el programador que utiliza la plantilla cuando aplica o añade su código a su propio sitio web.
  - c. Un campo que debe ser modificado por el creador de la plantilla cuando diseña la propia plantilla web.
  - d. Un campo que no debe ser modificado por el creador de la plantilla cuando diseña la propia plantilla web.
8. ¿Qué es un campo no modifiable o que no debe ser cambiado en una plantilla?
  - a. Un campo que nunca debe ser modificado por el programador que utiliza la plantilla cuando aplica o añade su código a su propio sitio web.
  - b. Un campo que debe ser modificado por el programador que utiliza la plantilla cuando aplica o añade su código a su propio sitio web.
  - c. Un campo que debe ser modificado por el creador de la plantilla cuando diseña la propia plantilla web.
  - d. Un campo que no debe ser modificado por el creador de la plantilla cuando diseña la propia plantilla web.
9. ¿Qué es un CMS o Sistema de Gestión de Contenidos?
  - a. Es un elemento de las plantillas especialmente diseñado para la creación de contenidos por el programador web.
  - b. Es un lenguaje de programación diseñado para la creación de sitios web.
  - c. Es un programa que permite, vía web, crear una estructura fija y visual para la creación y posterior administración de contenidos web.
  - d. Es una herramienta utilizada en internet para la creación interactiva en la nube de elementos web que luego pueden ser incorporados a un sitio web.

10. ¿Antes de realizar un cambio en una página web aplicando una plantilla encontrada en la red qué es lo más importante que se debe hacer?
- a. Una copia de seguridad.
  - b. Comprobar el código de la plantilla.
  - c. Comprobar el código del programa web.
  - d. Leerse las especificaciones de la plantilla.

# Glosario

- **ISP:** Es el proveedor de servicios de Internet (se le dice ISP por las siglas en inglés de Internet Service Provider). Es la empresa que proporciona conexión a Internet a sus clientes.
- **Servidor:** Un programa, aplicación informática o sistema informático que realiza algunas tareas en beneficio de otras aplicaciones o máquinas llamadas clientes.
- **URL:** Son las iniciales de localizador de recursos uniforme, (se le llama URL por las siglas en inglés de Uniform Resource Locator). Es una secuencia de caracteres que se utiliza para nombrar recursos en Internet para su identificación o localización.
- **Idempotente:** Es la propiedad para realizar una acción determinada varias veces y aun así conseguir el mismo resultado que se obtendría si se realizase una sola vez.
- **Unicode:** Es un estándar de codificación de caracteres que fue diseñado para facilitar el tratamiento informático, de transmisión y de visualización de textos de múltiples lenguajes, disciplinas técnicas y textos clásicos de lenguas muertas. El término proviene Universalidad, Uniformidad y Unicidad.
- **ASCII:** Es un acrónimo (que proviene de las iniciales en inglés de American Standard Code for Information Interchange) que identifica a un código de caracteres basados en el alfabeto latino, especialmente el que se usa en el inglés moderno y en otras lenguas occidentales.

- **ANSI:** Es el código de caracteres basados en el alfabeto latino del Instituto Nacional Estadounidense de Estándares (sus siglas en inglés vienen de American National Standards Institute), una organización sin ánimo de lucro que supervisa el desarrollo de estándares para productos, servicios, procesos y sistemas en los Estados Unidos.
- **CSS:** Son las hojas de estilo en cascada (en inglés sus iniciales son Cascading Style Sheets), que hacen referencia a un lenguaje de hojas de estilos usado para describir el aspecto y el formato de un documento escrito en lenguaje de marcas como HTML.
- **Hexadecimal:** Es un sistema de posición numeral con una base de 16, en lugar de la habitual base del sistema decimal de 10. Para representar los seis caracteres que faltan se usan las primeras letras del alfabeto.
- **RGB:** Es un modelo de color, llamado así por las iniciales en inglés de rojo, verde y azul, (Red, Green y Blue) basado en la síntesis aditiva. Es decir, que es posible representar un color mediante la mezcla por adición de los tres colores de luz primarios.
- **JavaScript:** Es un lenguaje de programación interpretado (se ejecuta por medio de un intérprete) que se utiliza principalmente en el lado del cliente, integrado en el código de un navegador web permitiendo mejoras en la interfaz de usuario y páginas web dinámicas.
- **Opera:** Es un navegador web creado por la empresa noruega Opera Software.
- **Clave Privada:** En el método criptográfico que usa un par de claves para el envío de mensajes se envían dos claves pertenecientes a la misma persona que ha enviado el mensaje. Una de ellas es la clave privada, que el propietario debe guardarla de modo que nadie tenga acceso a ella.
- **Clave Pública:** En el método criptográfico que usa un par de claves para el envío de mensajes se envían dos claves pertenecientes a la misma persona que ha enviado el mensaje. Una de ellas es la clave pública, que se puede entregar a cualquier persona.
- **Paquete:** Es cada uno de los bloques en que se divide, en el nivel de red y en el uso de internet, la información que se envía.
- **Algoritmo:** Una secuencia de instrucciones realizadas en el código de un lenguaje de programación que representan un modelo de solución para determinado tipo de problemas.

## Glosario

- **Script:** Es un programa que es normalmente simple y que normalmente se almacena en un archivo de texto plano, siendo casi siempre interpretado.
- **Maquetación:** Es una tarea encargada de organizar en un espacio, contenidos escritos, visuales y audiovisuales en medios impresos y electrónicos.
- **Opaco:** En informática este término se refiere a cuando uno de los lados no puede visualizar el código o el núcleo de un elemento informático, pero sí su ejecución.
- **Código Ofuscado:** Es aquel código creado en un lenguaje de programación que se ha realizado tan intrincado, enrevesado, enredado y confuso que resulta prácticamente imposible su descifrado incluso el programador que lo creó.
- **PHP:** Es un lenguaje de programación, donde se programa el código en el lado del servidor, originalmente diseñado y ampliamente utilizado para el desarrollo web de contenido dinámico.
- **CMS:** También llamado Sistema de Gestión de Contenidos (cuyas iniciales vienen del inglés Content Management System) es un programa que permite crear una estructura de soporte para la creación y administración de contenidos en páginas web.
- **Repositorio:** Es un sitio centralizado donde se guarda y se mantiene diferente información digital, habitualmente bases de datos o archivos informáticos.

Área: informática y comunicaciones

# Soluciones

## UF1304: Elaboración de plantillas y formularios

UD1	UD2
1. c	1. a
2. a	2. d
3. a	3. b
4. b	4. a
5. d	5. a
6. c	6. d
7. b	7. b
8. d	8. a
9. a	9. c
10. c	10. a

Área: informática y comunicaciones



