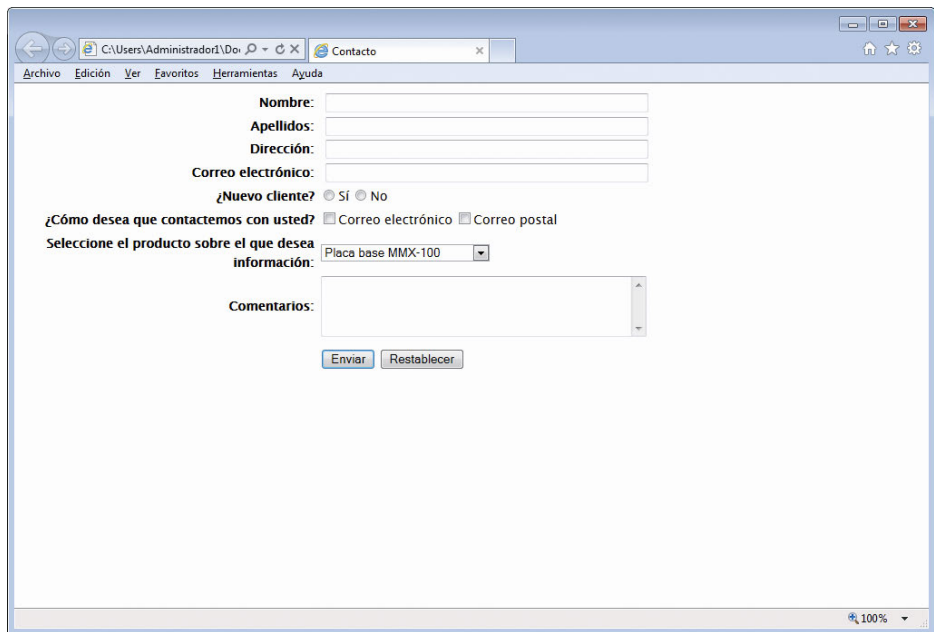


1. FORMULARIOS HTML

Los **formularios** son la forma más utilizada para recoger información del usuario en un sitio web.

El usuario rellena su información personal, establece algunas opciones y finalmente pulsa en un botón *Enviar* para que la información se transmita al servidor.

A screenshot of a web browser window showing a contact form. The browser's address bar displays 'C:\Users\Administrador\Do...' and the page title is 'Contacto'. The form includes several input fields: 'Nombre:', 'Apellidos:', 'Dirección:', and 'Correo electrónico:'. Below these is a radio button group for '¿Nuevo cliente?' with options 'Sí' and 'No'. A checkbox group for '¿Cómo desea que contactemos con usted?' includes 'Correo electrónico' and 'Correo postal'. A dropdown menu for 'Seleccione el producto sobre el que desea información:' shows 'Placa base MMX-100'. There is a large text area for 'Comentarios:'. At the bottom are two buttons: 'Enviar' and 'Restablecer'.

Esto lo podemos ver en la figura anterior. Se trata de un conjunto de elementos: botones, cuadros de texto, casillas de verificación, botones de opción, etc., que permiten que el usuario introduzca la información necesaria y la envíe al servidor web.

Los formularios se construyen utilizando etiquetas HTML. Sin embargo, el HTML solo proporciona la forma para crear el formulario, pero no para conseguir su funcionalidad.

Es decir, a la hora de procesar el envío de información al servidor, deberemos escribir una aplicación para ello, que se ejecutará en el servidor.

Vamos a repasar rápidamente el código HTML necesario para crear un formulario.

Para crear un formulario HTML, es necesario utilizar la etiqueta **<form>**. Esta etiqueta presenta los siguientes atributos:

- **name** e **id**: permite especificar el nombre del formulario. Es importante hacerlo, como veremos más tarde.

- **method:** especifica cómo se enviará la información del formulario al servidor. Hay dos métodos: GET y POST. El método más utilizado es POST y es el que normalmente se utilizará.
- **action:** especifica el nombre y ubicación de la aplicación que procesará la información. Como se ha indicado, esta aplicación se ejecuta en el servidor.
- **enctype:** indica el tipo de información que se envía.
- **target:** indica la ventana donde aparecerá la respuesta del servidor tras enviar la información. La respuesta puede ser la confirmación de que ha llegado bien o de que ha habido algún problema.

Entre las etiquetas `<form>` y `</form>` introduciremos las necesarias para crear los elementos del formulario, es decir, los cuadros de texto, botones, etc. Cada tipo de elemento tiene una etiqueta HTML asociada. Los que se han utilizado en el formulario anterior son:

- La etiqueta **input** sirve para cuadros de texto, botones, casillas de verificación, botones de opción y cuadros de contraseña. Utilizando el atributo **type**, se indica el tipo de dicho elemento.
- La etiqueta **select** sirve para crear una lista desplegable. Cada uno de los elementos de la lista se constituyen con la etiqueta **option**.
- Finalmente, utilizaremos la etiqueta `<textarea>` para crear un cuadro de texto de varias líneas, donde poder introducir mucha información.

Ya sabemos que los formularios se crean a partir de etiquetas HTML y que deberemos escribir una aplicación en la parte servidor para procesar el envío de la información.

Entonces, ¿qué papel juega JavaScript?

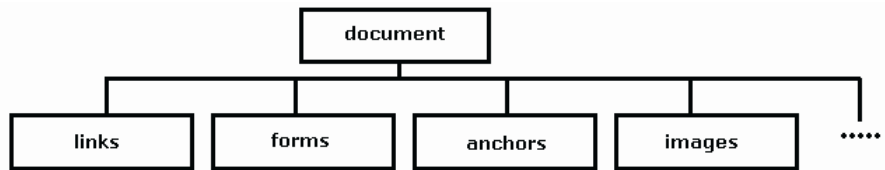
Podrás escribir un *script* para validar la información introducida en el formulario antes de que sea enviada al servidor. Esto es muy importante, ya que la información solo se envía tras haber superado la validación realizada en el equipo cliente, es decir, en el propio navegador.

El resto de la lección nos muestra cómo validar un formulario utilizando JavaScript.

2. EL CONJUNTO FORMS

Lo primero que debemos saber es que existe la propiedad **forms** del objeto **document**. Realmente, esta propiedad es un array de objetos form.

Esto quiere decir que cada formulario que introduzcas en la página web será un elemento del array forms. Así, la primera pareja de etiquetas `<form>...</form>` constituirá el elemento **document.forms[0]** y lo mismo con el resto de formularios que pueda tener la página web.



Aunque la notación que hemos utilizado normalmente para acceder a los elementos de un array es sencilla, podemos utilizar una forma alternativa de hacerlo, a través del identificador que hemos establecido para el formulario.

Para poder utilizar esta segunda notación, es necesario haber establecido el atributo **id** de la etiqueta `<form>`. El nombre indicado en dicho atributo es el utilizado en el código JavaScript.

En nuestro caso, hemos asignado al formulario el identificador "frmContacto", por lo que podremos utilizar las expresiones **document.forms[0]** o **document.frmContacto**.

El atributo **name** sigue siendo necesario, ya que lo utiliza el navegador a la hora de enviar la información al servidor web.

Sin embargo, en JavaScript lo importante es el atributo **id**, por lo que habitualmente tanto **id** como **name** tienen el mismo valor.

3. LA PROPIEDAD ELEMENTS

Sabemos cómo acceder a un determinado formulario utilizando el array **forms**. Pero normalmente, lo que haremos será acceder a los elementos del formulario: al contenido de los cuadros de texto, a la selección realizada en una lista desplegable, etc., para poder validar la información proporcionada por el usuario.



Es mejor y más cómodo utilizar el nombre del formulario, sobre todo si tenemos más de un formulario en la página web. Sin embargo, este no puede contener espacios ni caracteres como - , . :



Para ello, cada objeto **form** tiene una propiedad llamada **elements**. Como ya te imaginarás, esta propiedad es un array donde se almacenan los elementos del formulario.

Podrás hacer referencia a un determinado elemento bien utilizando el correspondiente índice en el array o bien a través del nombre indicado en su atributo **id**.

Es la misma idea que tenemos con el array **forms**. Así, para acceder al primer elemento, podremos utilizar estas cuatro expresiones:

```
document.forms[0].elements[0]
document.frmContacto.elements[0]
document.forms[0].txtApellidos
document.frmContacto.txtApellidos
```

Normalmente, cuando la información recogida en el formulario es de distinta naturaleza, será conveniente utilizar la notación en la que se incluye el nombre del elemento.

En otros casos, puede llegar a ser conveniente utilizar la notación con el índice del elemento. Por ejemplo, cuando deseemos aplicar un mismo proceso a todos los elementos del formulario a través de una estructura de repetición.

Debes saber que tanto el array **forms** como **elements** disponen de la propiedad **length**, que proporciona el número de elementos del array.

4. VALIDAR LA INFORMACIÓN

Vamos a aplicar toda esta teoría para escribir un *script* que valide alguna de las entradas introducidas por el usuario en este formulario.

Por ejemplo, vamos a comprobar que el usuario ha introducido el carácter @ (arroba) en el cuadro de texto referente a la dirección de correo electrónico. Este carácter aparece siempre en este tipo de direcciones.

```
function ValidarEmail(element)
{
    if (element.value.indexOf("@") == -1)
    {
        alert("Introduzca una dirección de correo electrónico válida.");
        element.focus();
        element.select();
    }
}
```

Utilizamos una función para validar lo que ha introducido el usuario en un elemento del formulario.

La función recibe dicho elemento a través del parámetro **element**, por lo que puede comprobar cuál es su contenido. Para ello utiliza la propiedad **value**.

Así, con la expresión **element.value**, lo que estamos haciendo es acceder al contenido introducido por el usuario en el elemento del formulario.

Recuerda que cada uno de los elementos del formulario es un objeto, por lo que puede tener sus correspondientes propiedades, métodos y eventos.

Utilizamos la función **indexOf** para comprobar el valor introducido en dicho cuadro de texto. Esta función devuelve el valor **-1** si el carácter que se pasa como parámetro no está en la cadena de texto sobre la que estamos aplicando la función.

Por ejemplo: el usuario introduce la cadena **jull@mail.com**. La función obtiene dicha cadena a través de la expresión **element.value** y le aplica la función **indexOf**, dando como resultado un número distinto de **-1**, ya que el carácter **@** está en la cadena de texto.

En el caso de que el resultado sea **-1**, mostraremos un cuadro de diálogo indicando que la dirección no es válida y aplicaremos los métodos **focus** y **select** al elemento. El primero le vuelve a dar el foco a dicho elemento, mientras que el segundo selecciona el texto que pueda tener.

De esta forma, facilitamos que el usuario sepa dónde ha cometido el posible error.



Los métodos **focus** y **select** no funcionan en el evento **blur** si se utiliza el navegador **Firefox**.

Parece que esto es una decisión de los diseñadores de Firefox. Sin embargo, en los otros navegadores populares sí que funciona.

Hay un “truco” para que funcione en Firefox. En lugar de utilizar:

```
element.focus();  
element.select();
```

utiliza

```
setTimeout(function(){element.focus()}, 10);  
setTimeout(function(){element.select()}, 10);
```

Por lo tanto, vemos cómo el elemento dispone de la propiedad **value** y de los métodos **focus** y **select**.

Acceder adecuadamente al elemento es la parte más difícil de entender. Una vez conseguido, utilizaremos las propiedades y métodos que necesitemos.

5. ¿CUÁNDO REALIZAR LA VALIDACIÓN?

Hablar de cuándo validar la información introducida en el formulario es lo mismo que estudiar algunos eventos interesantes.

En realidad tenemos dos opciones a la hora de decidir cuándo realizar la validación:

- Hacerlo a nivel de formulario, es decir, cuando el usuario decide enviar la información (por ejemplo, pulsando en un botón *Enviar*).
- Hacerlo a nivel de cada elemento del formulario, por ejemplo, cuando cambia su contenido.

En cualquier caso, debes recordar que la validación se realizará antes de enviar la información al servidor, que solo la recibirá si ha superado esta validación.

Por ejemplo, para validar la dirección de correo electrónico, podríamos escribir:

```
<input      name="txtEmail"      type="text"      class="campo  
cuadro_texto" id="txtEmail" onblur="ValidarEmail(this);">
```

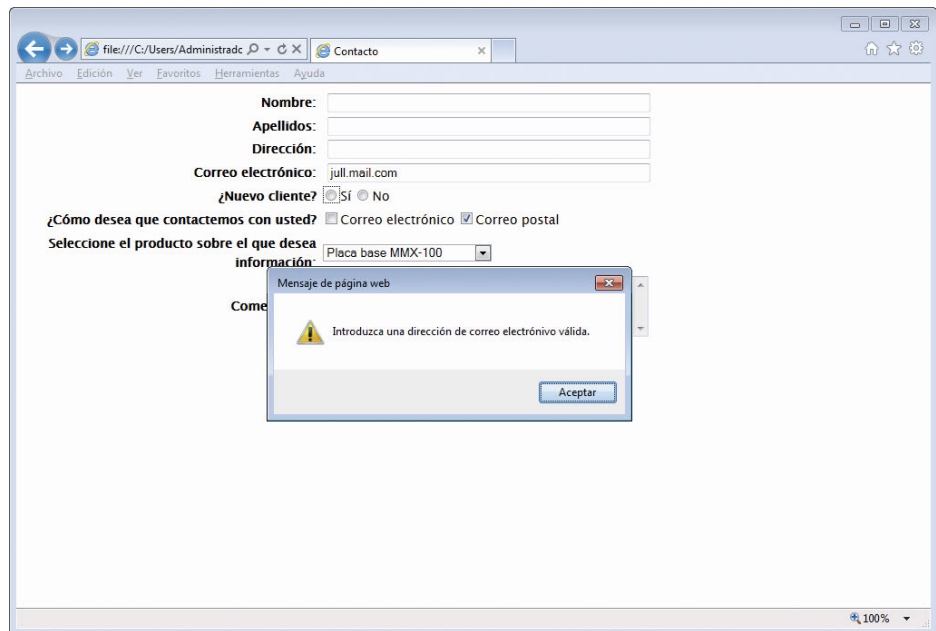
El evento utilizado ha sido **blur**. Se trata del evento que ocurre cuando el elemento pierde el foco.

Es decir, cuando el usuario se ha situado en el cuadro **txtEmail** y ha pasado a otro campo.

En el momento de situarse en el cuadro de texto **txtEmail**, este gana el foco, perdiéndolo cuando pasa a otro elemento del formulario.

En el evento **blur** decidimos llamar a la función **ValidarEmail**, pasándole como parámetro el propio elemento, es decir, el cuadro de texto **txtEmail**. Esto se consigue con la palabra reservada *this*.

En la figura de la página siguiente podemos ver el resultado.



Fíjate que la dirección incluida en el campo **Correo electrónico** no incluye el carácter @, por lo que no ha superado la validación.

El evento **blur** ocurre cada vez que el cuadro de texto pierde el foco. Otro evento que se podría utilizar es **change**.

En este caso, solo ocurre cuando el contenido del cuadro ha cambiado y tras perder el foco.

Si utilizas el evento **change**, la función debería devolver el valor *true* o *false* en función de si ha tenido éxito o no la validación realizada.

Aunque podemos utilizar el evento **blur** o **change**, normalmente esto es bastante molesto para el usuario, ya que puede repetirse el evento si no se corrige el valor introducido inmediatamente.

Si prefieres realizar la validación en el momento en que el usuario decide enviar la información, el evento a utilizar es **submit** del propio objeto formulario.

Por ejemplo, podríamos escribir una función que comprobara si se ha activado alguna de las casillas que permiten al usuario indicar cómo desea recibir la información: por correo postal o por correo electrónico:

```
function ValidarFormaContacto()
{
    var ok = false;
    if ((document.frmContacto.chkPorEmail.checked) ||
        (document.frmContacto.chkPorPost.checked))
        ok = true;
    if (!ok)
```

```
{
    alert("Tiene que indicar cómo desea que le
          enviemos la información.");
    return false;
}
else
    return true;
}
```

Es decir, utilizamos la variable **ok** para conocer si el formulario debe enviarse al servidor o no.

En este caso, nos basamos en si el usuario ha activado alguna (o las dos) de las casillas **chkPorEmail** y **chkPorPost** (esto se sabe a través de su propiedad **checked**).

En el caso de que no haya ninguna activada, **ok** valdrá **false** (que es su valor inicial); en el caso de que alguna (o las dos) esté activada, **ok** valdrá **true**.

Si, una vez comprobadas las casillas, **ok** sigue valiendo **false**, entonces avisamos al usuario de que debe elegir una forma de envío y cancelamos el envío al servidor devolviendo el valor **false**.

Fíjate que la expresión (**!ok**) es equivalente a (**ok == false**).

En caso contrario, la función devolverá el valor **true**, llevándose a cabo el envío del formulario.

Y así aplicaremos la función:

```
<form action="confirmacion.html" method="post"
name="frmContacto" id="frmContacto"
onsubmit="return ValidarFormaContacto();">
```

Utilizamos el evento **submit** (manejador **onsubmit**) para llamar a la función. Observa que esto producirá que se devuelva **true** o **false**, que es lo que requiere este evento.

Este evento corresponde al momento en que la información del formulario se envía al servidor web.

Así, si el código llamado en el evento **submit** se evalúa a **true**, la información es enviada al servidor; mientras que si se evalúa a **false**, entonces se cancela este envío al no haber sido superada la validación.

En la figura de la página siguiente puedes ver lo que ocurre cuando no se supera la validación. En este caso, la información del formulario no es enviada al servidor.

The screenshot shows a web browser window titled 'Contacto' with a file:// address. The browser's menu bar includes 'Archivo', 'Edición', 'Ver', 'Favoritos', 'Herramientas', and 'Ayuda'. The contact form contains the following fields and options:

- Nombre:** [Empty text box]
- Apellidos:** [Empty text box]
- Dirección:** [Empty text box]
- Correo electrónico:** [juli@mail.com]
- ¿Nuevo cliente?** ☐ Sí ☐ No
- ¿Cómo desea que contactemos con usted?** ☐ Correo electrónico ☐ Correo postal
- Seleccione el producto sobre el que desea información:** [Placa base MMX-100]

A modal dialog box titled 'Mensaje de página web' is displayed in the center, containing a yellow warning icon and the text: 'Tiene que indicar cómo desea que le enviemos la información.' An 'Aceptar' button is at the bottom right of the dialog.

Observa cómo la información no ha sido enviada, se mantiene en el formulario, ya que no ha superado la validación realizada, es decir, el evento **submit** ha devuelto el valor **false**.

Para que un botón actúe como un botón *Enviar*, es necesario que su tipo sea **submit**, independientemente del texto que en él se muestre.

Por su parte, un elemento de tipo **reset** es un botón *Restablecer*, es decir, que por defecto limpia la información introducida en el formulario. Si deseas programar algo al pulsar en este botón, deberías hacerlo en el evento **reset** (manejador **onreset**) del objeto formulario.

Si únicamente deseas esta funcionalidad, es suficiente con indicar que es un botón de tipo **reset**.

6. TIPOS DE VALIDACIÓN

En esta lección hemos visto cómo realizar la validación de los datos introducidos por el usuario en un formulario.

Esta validación se realiza antes de enviarse al servidor, gracias al código escrito en la parte cliente de la página web utilizando JavaScript.

Sin embargo, debe quedar muy claro qué tipo de validación se puede realizar de esta forma.

Así pues, podrás escribir código en la parte cliente para comprobar si una determinada dirección de correo electrónico es o no válida, es decir, si cumple los requisitos de cualquier dirección de este tipo.

Lo que no podrás hacer desde la parte cliente es validar si realmente esa dirección de correo electrónico existe en la base de datos de, por ejemplo, los clientes de la empresa.

Para esto último, tendrás que acceder a la base de datos, es decir, tendrás que crear una aplicación en la parte servidor del sitio web.

Además, la validación en el cliente no es segura, ya que el usuario podría desactivar el uso de JavaScript.

Por ello, independientemente de lo exhaustivo que sea el proceso de validación en el cliente, siempre se debe seguir realizando la validación en la aplicación del servidor web.