

1. INTRODUCCIÓN

Los **formularios** presentan bastantes novedades en HTML5. Estas novedades no solo incluyen nuevos controles de formulario, sino que HTML5 proporciona por primera vez la funcionalidad de validar los datos sin necesidad de recurrir a un lenguaje adicional como JavaScript.

Así pues, con HTML5 podremos hacer que un determinado campo sea requerido, que se ajuste al formato de una dirección de correo electrónico o que tenga un valor numérico dentro de un rango definido.

Todo esto indicándolo de una forma declarativa, es decir, a través de las etiquetas y atributos del lenguaje HTML5, sin necesidad de programar en JavaScript o en un lenguaje similar.

Esto no significa que la validación en el servidor ya no sea necesaria. Como siempre, debemos entender que la validación que realiza el navegador es un primer paso.

Adicionalmente, la aplicación del servidor realizará su correspondiente validación.

En cuanto a los navegadores modernos, **Opera** es el que más características de los formularios HTML5 implementa. Por eso lo vamos a utilizar en esta lección. Fíjate en la figura siguiente.

Formulario de pedido

Localhost/C:/Users/Administrador1/Documents/presentaciones/formulario_pedido1.html

Buscar con Google

Detalles personales

Nombre*:

Email*:

Teléfono*:

Dirección de envío

Dirección*:

Código postal*:

Ciudad*:

Provincia*:

* Campos obligatorios

Se trata de un formulario donde el usuario tiene que rellenar una serie de campos y pulsar en un botón. La estructura del formulario se ha conseguido con etiquetas tradicionales **HTML** y otras nuevas de **HTML5**.

El aspecto, mediante **CSS**. Por ahora no hay código **JavaScript** que se ejecute en este formulario.

Fíjate que algunos de los campos presentan un mensaje de ayuda. Por ejemplo, en el campo donde se debe introducir una dirección de correo electrónico, se indica el formato esperado.

Además, el navegador puede comprobar que hay campos obligatorios e indicarlo. Esta validación puede impedir que se envíe la información al servidor hasta haber completado correctamente el formulario.



The screenshot shows a form titled "Detalles personales" with three input fields: "Nombre*", "Email*", and "Teléfono". The "Email*" field contains the text "ejemplo@dominio.com" and has a red tooltip message that says "Este campo es necesario". The "Teléfono" field contains the text "Ej: +345555555".

Lo mismo puede ocurrir, por ejemplo, si se introduce una dirección de correo electrónica que no tiene el formato correcto.



The screenshot shows the same "Detalles personales" form. The "Nombre*" field now contains "Juan Rodríguez". The "Email*" field contains "jrodriguezmail.com" and has a red tooltip message that says "Escriba una dirección de correo válida". The "Teléfono" field contains "Ej: +345555555".

En la figura de la página siguiente se presenta otro formulario. Vemos que el campo **Noches** facilita la introducción de un valor numérico, así como el campo **Fecha entrada**, que permite elegir una determinada fecha directamente en un calendario.

Toda esta funcionalidad se ha conseguido únicamente mediante **HTML5**, sin necesidad de utilizar para nada **JavaScript**.

En esta lección conoceremos los nuevos elementos de HTML5 para formularios.

Además, veremos cómo "complementar" el código para que los navegadores antiguos que no entienden estas características, al menos presenten el formulario de forma que siga siendo útil para el usuario.

2. NUEVOS CONTROLES DE FORMULARIO

HTML5 incorpora algunos controles nuevos, como hemos podido comprobar anteriormente.

Estos controles tienen dos propósitos: desde el punto de vista semántico, la descripción adecuada de la información que recogen; por otra parte, facilitar la introducción de los distintos tipos de información y validarla si es el caso.

Para ello, HTML5 añade nuevos valores para el atributo **type** de la etiqueta **<input>**, introduciendo los siguientes controles especializados:

- **type="search"**: se trata de un cuadro de texto especializado en búsquedas. Es decir, que se utiliza para introducir palabras clave para realizar una búsqueda.

Realmente la diferencia entre el tradicional cuadro de texto (**type="text"**) y este nuevo es únicamente estético y cada navegador lo muestra de una forma distinta.

- **type="tel"**: cuadro de texto para representar números de teléfono.

Realmente no aporta demasiado este tipo de control, ya que no realiza ninguna validación adicional (de hecho, permite la introducción de letras y otros caracteres y no solo de números).

Algunos navegadores pueden proporcionar un teclado especial para la introducción de números de teléfono. Por ejemplo, el navegador de **iPhone** se personaliza para ello cuando el cursor se sitúa en un control de este tipo.

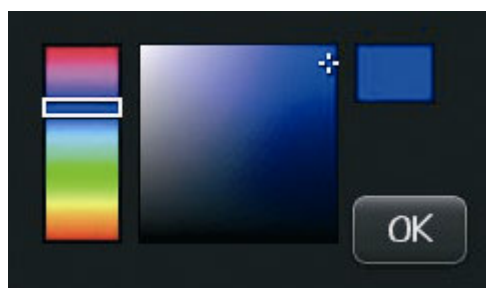


- **type="email"**: control especializado en la introducción de direcciones de correo electrónico.

Con este control el navegador es capaz de validar la información introducida por el usuario, comprobando que se trata de una dirección de correo electrónico válida.

Que exista o sea correcta ya lo decidirá la validación que realice la aplicación en el servidor.

- **type="color"**: es un control especializado en la introducción de colores. La figura siguiente muestra este control en un dispositivo **Blackberry**.





- **type="url"**: es un control especializado en la introducción de direcciones URL. Realiza la validación adecuada, de forma similar a la que realiza el tipo **email**.
- **type="number"**: control especializado en la introducción de valores numéricos. Solo admite números. En la figura siguiente, el campo *Noches* es de este tipo.



Si no se establece el valor del atributo **step**, entonces toma el valor 1.

Datos de la reserva

Nombre del hotel:

Noches*:  

Fecha entrada*:

Se puede afinar el valor esperado mediante los atributos **min** y **max**, que representan el valor mínimo y máximo, respectivamente, que puede introducir el usuario. Además, con el atributo **step**, indicamos cuánto se incrementa o decrementa cada vez.

En la figura anterior puedes ver que el navegador **Opera** proporciona para ello dos botones con una flecha hacia arriba y otra hacia abajo.

- **type="date"**, **type="time"**, **type="datetime"**, **type="datetime-local"**, **type="month"**, **type="week"**: controles especializados en la introducción de valores de fecha y hora.



Estos controles facilitan la introducción de fechas, que es un problema frecuente.

Así pues, el navegador puede proporcionar un calendario para que el usuario simplemente seleccione la fecha deseada; sin importarle cómo la representa internamente.

El navegador Opera utiliza un calendario para este tipo de control.



Datos de la reserva

Nombre del hotel:

Noches*:  

Fecha entrada*:

* Campos obligatorios

Diciembre2011 

Lun	Mar	Mie	Jue	Vie	Sab	Dom
28	29	30	1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	31	1
2	3	4	5	6	7	8

Hoy



Tenemos varios de estos tipos:

- **type="date"**: para introducir una fecha (mes, día y año).
- **type="time"**: para introducir una hora en el formato de 24 horas (hora, minutos y segundos) y con información de zona horaria.
- **type="datetime"**: para introducir una fecha y una hora.
- **type="datetime-local"**: igual que el anterior pero sin la posibilidad de indicar una zona horaria, ya que se entiende que es la zona local.
- **type="month"**: para introducir un mes de un año. Por ejemplo, el mes de diciembre de 2011 sería 2011-12.
- **type="week"**: para introducir una determinada semana en un año. Por ejemplo, la semana número 25 del año 2011 sería 2011-W25.

Todos estos tipos guardarán la fecha en el formato estandarizado: **AAAA-MM-DDThh:mm:ss.Z**

donde **A** (año), **M** (mes), **D** (día), **h** (hora), **m** (minuto), **s** (segundo) y **Z** (zona horaria).

Por ejemplo: **2011-04-27T17:03:21+01**

Además, admiten los atributos **min** y **max**, representando la fecha mínima y máxima que puede introducir el usuario.

- **type="range"**: para representar un control deslizante que se utiliza para introducir un valor numérico.

En este caso lo más importante no es el propio valor, sino la forma en la que se representa.



La ventaja de estos nuevos controles es que se representarán como un cuadro de texto tradicional si el navegador no los soporta. Es decir, como si el atributo **type** tuviera el valor **"text"**.

Por lo tanto, el formulario seguirá siendo válido, aunque mucho menos funcional, en un navegador antiguo.

Después veremos que es posible complementar el código de la página para "simular" esta funcionalidad mediante código **JavaScript**.

De esta forma, los navegadores modernos utilizarán la funcionalidad de HTML5 y los antiguos, la del código personalizado.

3. NUEVOS ATRIBUTOS

A parte de la incorporación de nuevos controles, la especificación de HTML5 también añade algunos atributos muy interesantes, en especial, el que permite realizar la validación de elementos requeridos.



Estos son los nuevos atributos HTML5 que vamos a estudiar:

- **required**
- **placeholder**
- **list** y el elemento **<datalist>**
- **autofocus**

Y es que ya no es necesario escribir pequeños programas en JavaScript para comprobar si el usuario ha introducido el valor en los campos que establecemos como **requeridos** u obligatorios.

Con HTML5 simplemente utilizaremos el atributo **required**.

Estudia el siguiente código:

```
<form id="frmPedido" action="#" method="post">
<fieldset>
<legend>Datos de la reserva</legend>
<ol>
  <li>
    <label for="txtNombreHotel">Nombre del hotel:
    </label>
    <input id="txtNombreHotel" name="txtNombreHotel"
      type="text" value="Hotel Experia II" readonly>
    </li>
  <li>
    <label for="txtNumeroNoches">Noches*:</label>
    <input id="txtNumeroNoches" name="txtNumeroNoches"
      type="number" min="1" max="10" required>
    </li>
  <li>
    <label for="txtFechaEntrada">Fecha entrada*:
    </label>
    <input id="txtFechaEntrada" name="txtFechaEntrada"
      type="date" required>
    </li>
</ol>
</fieldset>
<fieldset>
  <input type="submit" value="Finalizar">
</fieldset>
<p><small>* Campos obligatorios</small></p>
</form>
```

Fíjate en la línea donde se añade el control correspondiente al campo **Noches**. Como puedes ver, se utiliza el atributo **required**. Lo mismo podemos decir en la línea para el campo de la fecha de entrada.

Simplemente con utilizar este atributo, el navegador se encargará de asegurar que el usuario introduce un valor en esos campos antes de enviar la información al servidor.

Esta será la primera validación. Después podría completarse con otras. Por ejemplo, para un control de tipo **email** y **requerido**, primero se comprobará que el usuario ha introducido un valor y, después, que dicho valor es una dirección de correo electrónico válida.

Otro atributo cuyo resultado hasta ahora conseguíamos mediante JavaScript es **placeholder**.

El atributo **placeholder** permite introducir texto de ejemplo o de consejo en los cuadros de texto. La idea es que el usuario reciba alguna indicación para rellenar el campo con la información solicitada.

Por ejemplo, hemos utilizado este atributo en los cuadros de texto de la parte superior, donde se solicita el nombre, email y teléfono del usuario.

Así pues, alguna indicación para que el usuario sepa que debe introducir su nombre y apellidos o el formato que se espera para su teléfono pueden ser de gran ayuda.

Detalles personales

Nombre*: Nombre y apellidos

Email*: ejemplo@dominio.com

Teléfono: Ej: +345555555

Fíjate en el código correspondiente:

```
<fieldset>
<legend>Detalles personales</legend>
<ol>
  <li>
    <label for="txtNombre">Nombre*:</label>
    <input id="txtNombre" name="txtNombre" type="text"
      placeholder="Nombre y apellidos" required autofocus>
  </li>
  <li>
    <label for="txtEmail">Email*:</label>
    <input id="txtEmail" name="txtEmail" type="email"
      placeholder="ejemplo@dominio.com" required>
  </li>
</ol>
```



```

<li>
<label for="txtTelefono">Teléfono:</label>
<input id="txtTelefono" name="txtTelefono"
  type="tel" placeholder="Ej: +345555555">
</li>
</ol>
</fieldset>

```

El valor del atributo **placeholder** es una cadena con el texto de ayuda.

Cuando el usuario pulsa en uno de estos campos, ese texto desaparece, permitiendo que introduzca sus datos.

Por su parte, el atributo **list** permite mostrar una lista asociada a un cuadro de texto. De esta forma, el usuario no tiene que escribir el valor deseado (aunque puede hacerlo e incluso escribir uno distinto a los que aparecen en la lista) sino simplemente seleccionar un elemento de la lista.

El valor del atributo **list** es el identificador de una lista que seguidamente se define con la etiqueta **<datalist>**, que también es nueva en HTML5.

```

<label for="txtProvincia">Provincia:</label>
<input type="text" name="txtProvincia"
  id="txtProvincia" list="provincias">
<datalist id="provincias">
<option label="Álava" value="Álava">
<option label="Albacete" value="Albacete">
<option label="Alicante" value="Alicante">
<option label="Almería" value="Almería">
<option label="Asturias" value="Asturias">
</datalist>

```

Observa el resultado obtenido en el navegador Opera.

The screenshot shows a form titled "Dirección de envío" with the following fields:

- Dirección*:** A text input field.
- Código postal*:** A text input field.
- Ciudad:** A text input field.
- Provincia:** A dropdown menu with a list of Spanish provinces: Álava, Albacete, Alicante, Almería, and Asturias.

Como puedes ver, es muy parecido a un control **<select>**, pero utilizando dicha funcionalidad con un cuadro de texto.

Y, finalmente, podemos encontrar el atributo **autofocus**. Con este atributo podemos hacer que tome el foco un determinado control.



Por ejemplo, la página principal del buscador **Google** está preparada de forma que, al cargar la página, tome el foco el campo de búsqueda donde introducimos las palabras clave de la búsqueda.



Solo debe haber un control con el atributo **autofocus** en una página web.

Con HTML5 esto es tan sencillo como establecer el valor **autofocus** para dicho control. Esto no está limitado a los controles **input**, también podremos utilizarlo con controles **textarea** o **select**.

Este atributo debe utilizarse con precaución, ya que puede producir resultados indeseados. Por ello, utilízalo únicamente cuando tenga sentido.

Además, recuerda que todas estas nuevas características están soportadas de distinta manera por los navegadores. La especificación HTML5 da bastante libertad a la hora de la implementación.

4. COMPATIBILIDAD CON NAVEGADORES ANTIGUOS

Bueno, ¿y qué ocurre con aquellos navegadores que no entienden las nuevas características de los formularios HTML5?

Pues si queremos ofrecer la misma funcionalidad, tendremos que complementar el código HTML5. Eso sí, mediante un lenguaje auxiliar de *scripting*, como **JavaScript**.

Así que si ya estabas pensando en eliminar todo el código JavaScript que tienes en tus páginas web para validar los datos introducidos en un formulario, mejor será que no lo hagas todavía.

La estrategia que vamos a seguir es esta:

1.- Comprobar si el navegador admite un elemento o atributo HTML5.

1.1.- En caso positivo, no hacer nada más, simplemente incluir el código HTML5.

1.2.- En caso negativo, ejecutar el código adicional JavaScript.

Fíjate que no comprobaremos qué navegador está utilizando el usuario, sino si es compatible con las características HTML5 que hemos utilizado.

Lo veremos con un ejemplo: el atributo **placeholder**. Estudia el código JavaScript siguiente:

```
function elementSupportsAttribute(element, attribute) {
    var test = document.createElement(element);
    if (attribute in test) {
        return true;
    } else {
        return false;
    }
}
```

Se trata de una función que sirve para comprobar si el navegador admite una determinada característica HTML5.

Fíjate cómo lo hacemos:

- Se crea un elemento en la memoria del navegador, pero sin incluirlo en el DOM del documento:

```
var test = document.createElement(element);
```

- Se testea si el elemento acepta el atributo en cuestión. Si es así, se devuelve el valor **true**; en caso contrario, el valor **false**.

Recuerda que el elemento que creamos para probar la característica no forma parte del DOM del documento, por lo que realmente no aparecerá de ninguna forma en la página web.



Alternativamente, puedes utilizar la librería **Modernizr** (www.modernizr.com). Se trata de una librería que proporciona métodos para comprobar la compatibilidad del navegador del usuario con las características de HTML5.

Por ejemplo, la expresión **Modernizr.inputtypes[*email*]** se evaluará a **true** si el navegador es compatible con el tipo de control **email** y **false** en caso contrario.

Sin embargo, esta librería no aporta ningún código alternativo para que los navegadores antiguos obtengan la misma funcionalidad que no entienden del lenguaje HTML5, aunque en el sitio web www.modernizr.com encontrarás enlaces a otros lugares donde encontrar dicho código.

Veamos ahora cómo utilizamos esta función:

```
var nombre = document.getElementById("txtNombre");
var email = document.getElementById("txtEmail");
var tel = document.getElementById("txtTelefono");

window.onload = function() {
  if (!elementSupportsAttribute("input", "placeholder"))
  {
    nombre.setAttribute("style", "color: gray;");
    email.setAttribute("style", "color: gray;");
    tel.setAttribute("style", "color: gray;");
    nombre.value = nombre.getAttribute("placeholder");
    email.value = email.getAttribute("placeholder");
    tel.value = tel.getAttribute("placeholder");
  }
}
```

En estas líneas encontramos el código que se ejecuta al cargar la página. Por eso está asociado con el evento **window.onload**.

Fíjate que se comprueba si el navegador es compatible con el atributo **placeholder** y, en caso negativo, se realizan los pasos oportunos para simular esta característica mediante JavaScript.

En este caso se establece el valor de los atributos **style** y **value** para mostrar el texto correspondiente en los campos de nombre, email y teléfono.

Fíjate que dicho texto se puede obtener del propio atributo **placeholder** de la etiqueta **<input>**. Es decir, que aunque el navegador no es compatible con dicho atributo, sí que podemos obtener su valor a través de JavaScript.

Sin entrar en mayores detalles, aquí vemos que si el navegador es compatible con la característica HTML5, no se ejecutará ningún código JavaScript; si no es compatible, se ejecutará el código JavaScript para simular dicha característica.

De forma similar, actuaríamos con otras características HTML5 que queremos hacer compatibles con los navegadores antiguos.

Sin embargo, es importante darse cuenta de que la página sigue siendo válida incluso sin esta funcionalidad. Y es que uno de los objetivos del diseño de HTML5 es ser compatible "*hacia atrás*", es decir, con aquellos navegadores que no entienden las nuevas características.

Por eso, estos navegadores interpretan como un cuadro de texto tradicional (**type="text"**) los controles nuevos que hemos visto durante la lección.

A medida de que los navegadores antiguos sean reemplazados por los modernos, todo este código adicional JavaScript no será necesario.

