

# HTML5

- Incorpora nuevas **etiquetas de carácter semántico** y que nos ayudan a definir la estructura. Por ejemplo `<footer>`, para el pie de página.
- Nuevos controles para formularios que antes sólo eran posibles con JavaScript o [CSS](#).
- Soporta audio y vídeo de forma nativa.
- Ya no es necesario el uso de comillas dobles en los atributos, siendo posible algo como: `<div id=contenedor>Esto es un div</div>`.
- No hay una sintaxis tan estricta y no es necesario cerrar las etiquetas vacías, por lo que `<br>` sería válido.
- Etiqueta *doctype* simplificada.
- Podemos dibujar con etiquetas como `<canvas>`.
- Se han eliminado etiquetas y atributos no necesarios como los `<frames>`.
- Ahora las etiquetas `html`, `head` y `body` no son obligatorias.
- Las etiquetas de tablas `thead`, `tbody` y `tfoot` tampoco son obligatorias.

Aquí podemos ver la [documentación de W3C](#) sobre la última versión publicada, actualmente la 5.2.

## 1. Declaración DOCTYPE

Hasta ahora no hemos hablado de un concepto de vital importancia que es la utilización de elementos [HTML](#) estándares.

Con las diferentes versiones de HTML, algunas etiquetas dejan de tener uso o aparecen otras nuevas.

La organización que define los estándares para la web es [W3C](#).

Utilizar en lo posible las directivas de este comité de estándares nos traerá como ventaja que nuestras páginas en un futuro sigan viéndose correctamente en las nuevas versiones de navegadores.

La versión más actual de HTML es la 5.

En algunos de los documentos hemos introducido la declaración del tipo de documento (DTD Document Type Declaration), esta sección se ubica en la primera línea del archivo HTML, es decir antes de la marca `html`.

En HTML5 se ha simplificado mucho, quedando de la forma:

```
<!DOCTYPE html>
```

En la página [w3schools](#) podemos ver cómo eran las versiones anteriores.

## 2. Estructura de documentos HTML5

Veamos las nuevas etiquetas que nos permiten estructurar nuestro contenido en HTML5 más allá del clásico `<div>`:

- **<header>** No confundir con *head*. Pensada para albergar la información de la parte superior de la página o cabecera. Logos, títulos, descripciones, formularios de búsqueda, etc. Se puede usar tantas veces como se quiera creando de este modo secciones.

```
<header>
  <h1>Mi restaurante</h1>
  <nav>
    <ul>
      <li><a href="" class="current">Inicio</a></li>
      <li><a href="">clases</a></li>
      <li><a href="">catering</a></li>
      <li><a href="">Sobre mí</a></li>
      <li><a href="">Contacto</a></li>
    </ul>
  </nav>
</header>
```

- **<footer>** Para el pie de página. Esta sección va al final y suele incluir información como los datos de contacto, ayuda, textos legales, etc. También se puede usar varias veces para crear secciones.

```
<footer>
  &copy; 2018 Diego C Martín
</footer>
```

- **<nav>** Para incluir el menú principal de la página, que nos de acceso al resto de páginas principales de un sitio web. Se puede usar también tantas veces como se quiera, pero es mejor usarla sólo una vez por página.

```
<nav>
  <ul>
    <li><a href="" class="current">Inicio</a></li>
    <li><a href="">clases</a></li>
    <li><a href="">catering</a></li>
    <li><a href="">Sobre mí</a></li>
    <li><a href="">Contacto</a></li>
  </ul>
</nav>
```

- **<article>** Para incluir información que podría tener sentido de forma independiente, como un artículo de blog o noticia. Podemos incluir dentro de ella *header* y *footer*.

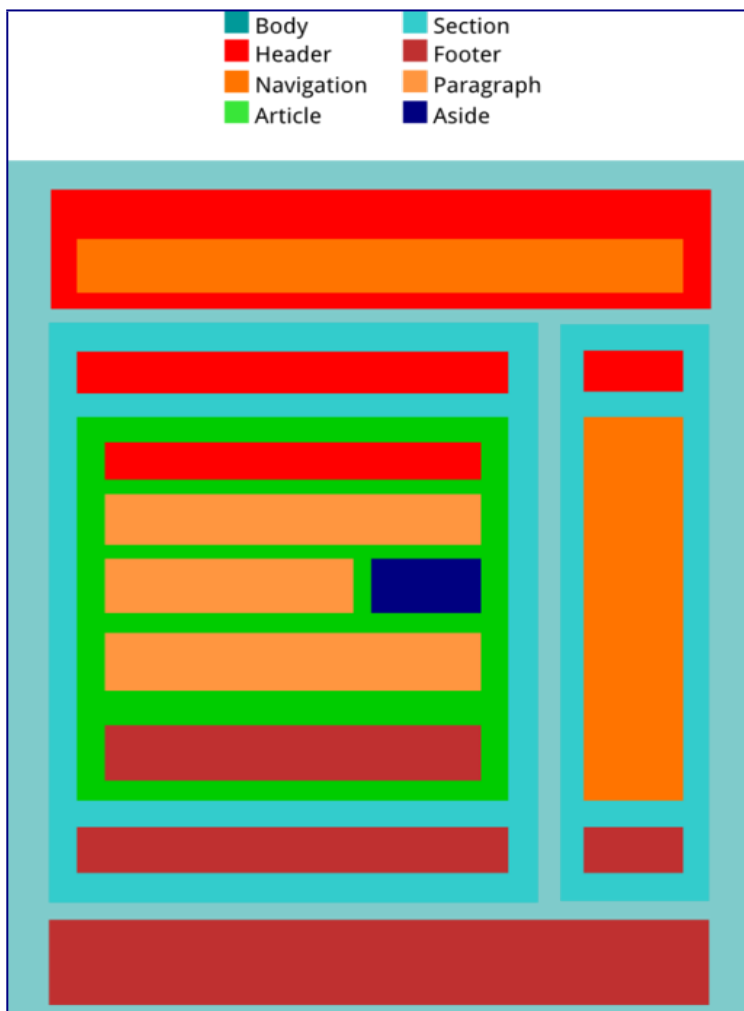
```
<article>
  <h2>Título del artículo 1</h2>
  <h3>subtítulo del artículo 1</h3>
  <p>Contenido del artículo 1. Esto pueden ser muchos párrafos.</p>
</article>
<article>
  <h2>Título del artículo 2</h2>
  <h3>subtítulo del artículo 2</h3>
  <p>Contenido del artículo 2. Esto pueden ser muchos párrafos.</p>
</article>
```

- **<section>** Para crear secciones y agrupar artículos por temática por ejemplo.

```
<section class="recetasjaponesas">
  <h2>Recetas japonesas</h2>
  <a href="">Pollo Yakitori</a>
  <a href="">Pollo Tsukune</a>
  <a href="">Okonomiyaki</a>
  <a href="">Mizutaki</a>
</section>
<section class="contacto">
  <h2>Contacto</h2>
  <p>Mi restaurante<br />
  21 Redchurch Street<br />
  Shoreditch<br />
  London E2 7DP</p>
</section>
```

- **<aside>** Para contenido de carácter secundario y que visualmente suele ir ubicado en un lateral de forma que guarde relación con toda la página o bien dentro de un artículo. Ponemos enlaces relacionados, banners publicitarios, etc. Un ejemplo sería igual al anterior pero intercambiando la etiqueta *section* por *aside*.

Veamos un ejemplo de estructura que podríamos crear:



## Atributo *Role* en HTML5

También podemos usar la etiqueta `<main>` para especificar el contenido principal de la página.

Además, contamos con el atributo *role*, al cual podemos asignar los siguientes valores:

- *main*
- *secondary*, parte secundaria del documento.
- *navigation*
- *banner*, banners, logos, etc.
- *contentinfo*, para elementos que aportan información sobre el contenido de la página (autores, copyrights, legal...)
- *definition*, definiciones.
- *note*, notas adicionales.
- *seealso*, para información relacionada.
- *search*, para formularios de búsqueda.

Al asignar roles nos aseguramos que las web sean mucho más legibles para usuarios discapacitados. Por ejemplo, la etiqueta *main* anteriormente mencionada, debería llevar el atributo *role* con el valor *main*, *nav*, con *navigation*, etc. Ejemplo:

```
<div id="buscar" role="search">
...
</div>
```

## 3. Elementos multimedia HTML5

### Audio

Usaremos la etiqueta `<audio>`, en la cual debemos incluir `<source>` para especificar la ubicación y característica de los ficheros mediante los atributos *src* y *type*.

En la etiqueta audio podemos usar atributos como *autoplay*, *controls*, *loop* o *muted*.

Estos atributos son booleanos, por tanto, no es necesario que especifiquemos valor:

```
<audio controls autoplay>
  <source src="media/cancion.mp3" type="audio/mpeg">
</audio>
```

### Video

Es casi igual que audio, salvo que en `<video>` podemos especificar los atributos *width* y *height*.

```
<video controls>
  <source src="media/peli.mp4" type="video/mp4">
  <source src="media/peli.ogv" type="video/ogv">
</video>
```

## Formatos de audio y vídeo

Nombre	Formato	Tipo MIME	IE 9+	Chrome	Firefox 4+	Safari 5+	Opera 10+
Audio mp3	mp3	audio/mpeg	x	x		x	
Audio Ogg	ogg	audio/ogg		x	x	x	x
Audio Wav	wav	audio/wav		x	x		x
Vídeo mp4	mp4	video/mp4	x	x		x	
Vídeo WebM	webm	video/webm		x	x		x
Vídeo Ogg	ogv	video/ogv		x	x		x

## Recursos para material multimedia

Biblioteca y alojamiento de material multimedia en [archive.org](http://archive.org).

[Software](#) libre y multiplataforma de edición de audio: [Audacity](#).

Software libre y multiplataforma de edición de vídeo: [Kdenlive](#).

## 4. Nuevos controles de formulario

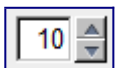
Como los formularios son la principal herramienta para la introducción de datos en las aplicaciones web y los datos que deseamos recopilar se han vuelto más complejos, ha sido necesario crear un elemento input con más capacidades, para recolectar estos datos con más semántica y una mejor definición, además de permitir un más fácil y eficaz manejo de errores y validación.

### `<input type="number">`

El primer nuevo tipo de campo de entrada que analizaremos es `type="number"`:

`<input type="number" ... >`

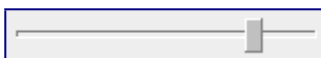
Este crea un tipo especial de campo de entrada para la introducción de un número. En la mayoría de los navegadores que lo soportan, aparece como un campo de entrada de texto con un control de número, que le permite aumentar o disminuir su valor.



**Figura 2:** Un campo de entrada de tipo *number*.

### `<input type="range">`

Crear un control deslizante que le permita elegir entre un rango de valores solía ser una propuesta complicada, semánticamente dudosa, pero con el HTML5 es fácil: sólo tiene que utilizar el tipo de campo de entrada `range`: `<input type="range" ... >`



**Figura 3:** Un campo de entrada *range*.

Tenga en cuenta que, por defecto, este campo de entrada no suele mostrar el valor seleccionado en el momento actual, o incluso el rango de valores que abarca. Los desarrolladores deben proporcionarlos mediante otros medios; por ejemplo, para mostrar el valor actual, podríamos utilizar un elemento `<output>` junto con algo de JavaScript para actualizar la representación del control cuando el usuario ha interactuado con el formulario:

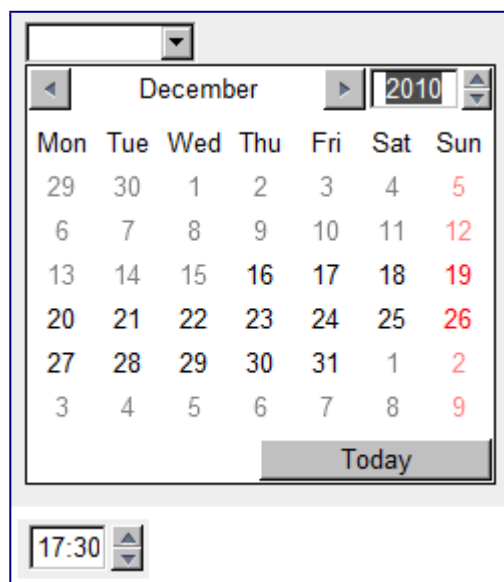
```
<output onforminput="value=weight.value"></output>
```

## **`<input type="date">` y otros controles de fecha/hora**

HTML5 tiene un número de campos de entrada diferentes para crear selectores complejos de fecha/hora; por ejemplo, el tipo de selector de fecha que aparece en prácticamente todo sitio de reserva de vuelo/tren existente. Estos suelen crearse utilizando trucos no semánticos, por lo que es estupendo que ahora tengamos formas estandarizadas fáciles para hacerlo. Por ejemplo:

```
<input type="date" ... >  
<input type="time" ... >
```

Estos crean, respectivamente, un selector de fecha totalmente funcional y un campo de entrada de texto que contiene un separador para horas, minutos y segundos (dependiendo del atributo `step` especificado) que sólo le permite introducir un valor de tiempo.



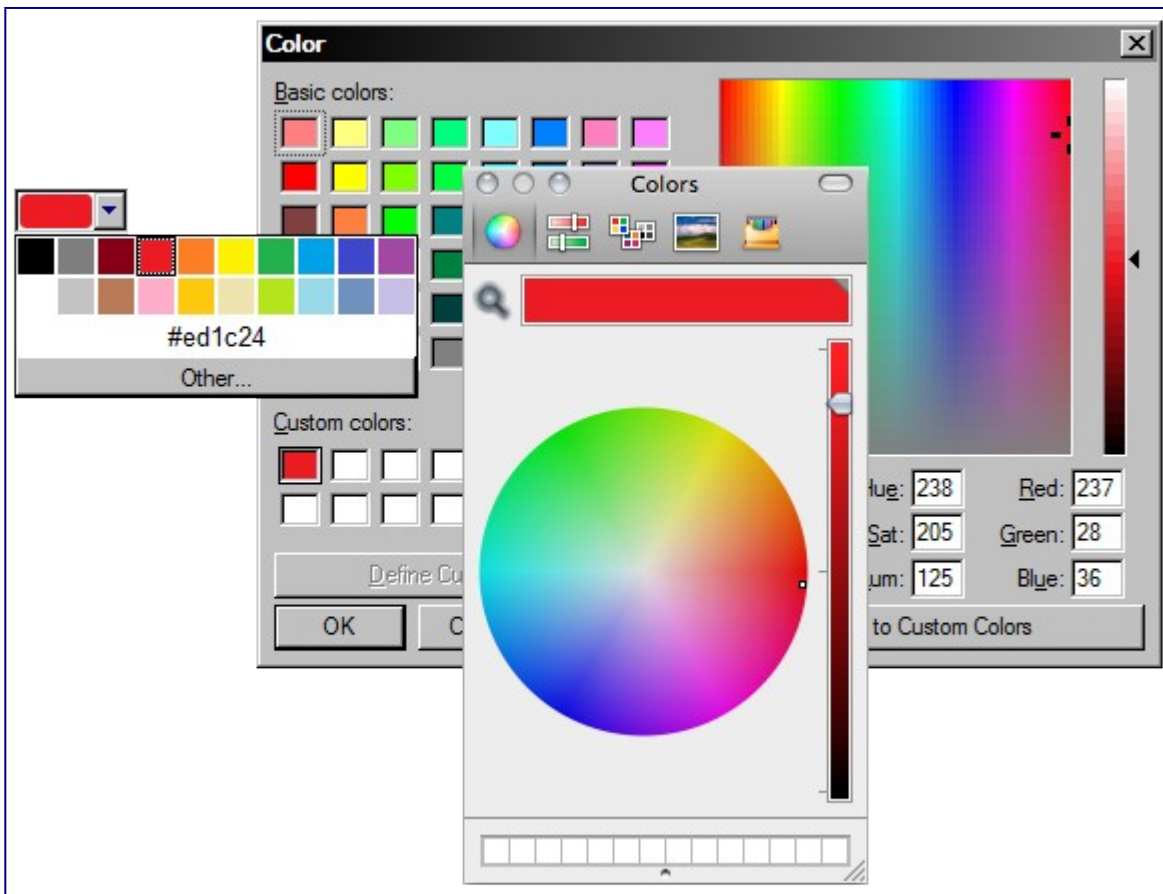
**Figura 4: Campos de entrada `date` y `time`.**

Pero allí no termina todo — hay un número de otros campos de entrada relacionados disponibles:

- **`datetime`**: combina la funcionalidad de los dos que hemos analizado anteriormente, permitiéndole elegir una fecha y una hora.
- **`month`**: le permite elegir un mes, almacenado internamente como un número entre 1-12; aunque los diferentes navegadores pueden proporcionarle mecanismos de selección más elaborados, como una lista desplegable con los nombres de los meses.
- **`week`**: le permite elegir una semana, almacenada internamente en el formato 2010-W37 (semana 37 del año 2010) y elegida mediante un selector de fecha similar a los que ya hemos visto.

## **<input type="color">**

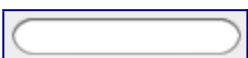
Este campo de entrada muestra un selector de color. La implementación de Opera le permite al usuario elegir entre una selección de colores, introducir valores hexadecimales directamente en un campo de texto o invocar el selector de colores nativo del Sistema Operativo.



**Figura 5:** un campo de entrada *color* y los selectores de color nativos de Windows y OS X.

## **<input type="search">**

El campo de entrada `search` podría decirse que no es nada más que un campo de entrada de texto con un estilo diferente. Los navegadores deberían aplicarle a estos campos de entrada el mismo estilo que a cualquier funcionalidad de búsqueda específica del sistema operativo. Sin embargo, más allá de esta consideración puramente estética, sigue siendo importante tener en cuenta que el etiquetar explícitamente campos de búsqueda abre la posibilidad para los navegadores, ayudas técnicas o rastreadores automatizados, de hacer algo inteligente con estas entradas en el futuro; por ejemplo, un navegador podría, posiblemente, ofrecer al usuario una opción para crear automáticamente una búsqueda personalizada para un sitio específico.



**Figura 6:** Un campo de entrada *search* como se ve en Opera en OS X.

## El elemento <datalist> y el atributo list

Hasta ahora hemos estado acostumbrados a utilizar los elementos <select> y <option> para crear listas desplegables de opciones donde nuestros usuarios pueden elegir. Pero, ¿y si quisiéramos crear una lista que le permita a los usuarios elegir entre una lista de opciones sugeridas, así como la posibilidad de introducir su propia opción? Esto solía requerir secuencias de comandos complicadas, pero ahora puede simplemente usar el atributo `list` para conectar un campo de entrada corriente a una lista de opciones, definidas dentro de un elemento <datalist>.

```
<input type="text" list="misdatos" ... >
<datalist id="misdatos">
  <option label="Sr." value="Señor">
  <option label="Sra." value="Señora">
  <option label="Srta." value="Señorita">
</datalist>
```

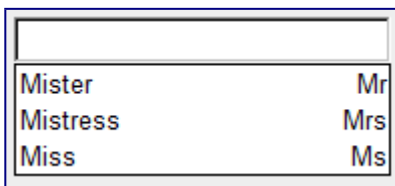


Figure 7: Creación de un campo de entrada con sugerencias usando `datalist`.

## <input type="tel">, <input type="email"> e <input type="url">

Como sus nombres lo indican, estos nuevos campos de entrada se refieren a números de teléfono, direcciones de correo electrónico y URLs. Los navegadores los mostrarán como campos de entrada de texto normales, pero señalar claramente qué tipo de texto estamos esperando en estos campos juega un papel importante en la validación de formularios del lado del cliente. Además, en ciertos dispositivos móviles el navegador cambiará su teclado en pantalla para entrada de texto común por sus variantes más relevantes al contexto. Una vez más, es concebible que en el futuro los navegadores aprovecharán mejor estas entradas explícitamente definidas para ofrecer funcionalidades adicionales, como el autocompletado de direcciones de correos electrónicos y números telefónicos basándose en la lista de contactos o la libreta de direcciones del usuario.

## Nuevos atributos

Además de nuevos campos de entrada explícitos, el HTML5 define una serie de nuevos atributos para los controles de formulario que ayudan a simplificar algunas tareas comunes y precisar los valores esperados para determinados campos de entrada.

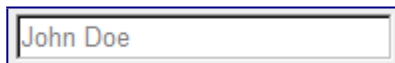
### placeholder

Un truco común de usabilidad para los formularios web es tener un contenido predeterminado en los campos de entrada de texto; por ejemplo, para dar sugerencias acerca del tipo de información que esperamos que el usuario introduzca, y que desaparecerá cuando dicho control de formulario reciba el foco del usuario.



Aunque esto solía requerir algo de JavaScript (borrar el contenido del campo de formulario al recibir el foco y restablecer el texto predeterminado si el usuario retira el foco del campo sin introducir nada), ahora podemos simplemente utilizar el atributo `placeholder`:

```
<input type="text"... placeholder="John Doe">
```

A screenshot of a text input field. The field is rectangular with a thin border. Inside the field, the text "John Doe" is displayed in a light gray font, which is the placeholder text. The field is currently empty of user input.

**Figura 8:** Un campo de entrada de texto con texto predeterminado mediante el uso de *placeholder*.

## autofocus

Otra característica común que anteriormente dependía de secuencias de comandos es la colocación automática del foco en un campo de formulario cuando se carga una página. Esto puede lograrse ahora con el atributo `autofocus`:

```
<input type="text" autofocus ... >
```

Tenga en cuenta que no debe tener más de un control de formulario con el atributo `autofocus` en una misma página. Además, debe utilizar este tipo de funcionalidad con cautela, en situaciones donde un formulario representa la principal área de interés en una página. Una página de búsqueda es un buen ejemplo, siempre que no haya mucho contenido y texto descriptivo, tiene sentido establecer el foco automáticamente en el campo de entrada de texto del formulario de búsqueda.

## min y max

Como su nombre sugiere, este par de atributos le permiten establecer un límite inferior y superior para los valores que se pueden introducir en un campo de entrada numérico; por ejemplo, campos de entrada de número, rango (control deslizante), hora o fecha (sí, incluso se puede utilizar para establecer los límites superior e inferior de las fechas, por ejemplo, en un formulario de reserva de vuelos podría limitar el selector de fechas para que sólo permita que el usuario seleccione fechas futuras). Para campos de entrada `range`, `min` y `max` son realmente necesarios para definir los valores que se devuelven cuando se envía el formulario. El código es bastante simple y autoexplicativo:

```
<input type="number" ... min="1" max="10">
```

## step

El atributo `step` puede utilizarse con un valor de campo de entrada numérico para establecer la precisión de los valores que se pueden ingresar. Por ejemplo, puede que desee que los usuarios introduzcan un tiempo determinado, pero sólo en incrementos de 30 minutos. En este caso, podemos utilizar el atributo `step`, teniendo en cuenta que para campos de entrada `time` el valor de este atributo se establece en segundos:

```
<input type="time" ... step="1800">
```

## Nuevos mecanismos de salida

Más allá de los nuevos controles de formulario con los que los usuarios pueden interactuar, HTML5 define una serie de nuevos elementos específicamente destinados a mostrar y visualizar información para el usuario.

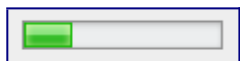
### <output>

Ya hemos mencionado el elemento <output> cuando hablamos del campo de entrada `range`. Este elemento sirve como una forma de mostrar el resultado de un cálculo, o más generalmente, para proporcionar una salida explícitamente identificada a una secuencia de comandos (en lugar de simplemente colocar un texto dentro de un `span` o `div` al azar). Para hacer aún más explícito a que controles de formulario en particular está vinculado el <output>, podemos, de una manera similar al elemento <label>, pasar una lista de IDs en el atributo opcional `for` del elemento.

```
<input type="range" id="ejemplorango" ... >
<output onforminput="value=ejemplorango.value" for="ejemplorango"></output>
```

### <progress> y <meter>

Estos dos nuevos elementos son muy similares. Ambos muestran un indicador/barra al usuario como resultado. Lo que los distingue es su propósito. Como su nombre sugiere, <progress> tiene el fin de representar una barra de progreso para indicar el porcentaje de finalización de una tarea determinada, mientras que <meter> es un indicador más genérico de una medida escalar o valor fraccionario.



*Figura 9: Una barra indicadora de progreso.*

## Validación

La validación de formularios es muy importante tanto en el lado cliente, como en el lado servidor, para ayudar a los usuarios reales a evitar y corregir los errores y para evitar que usuarios maliciosos envíen datos que pudieran causar daños a nuestra aplicación. Dado que los navegadores pueden ahora tener una idea de qué tipo de valores se esperan para los diversos controles de formulario (ya sea mediante su `type`, o cualquier límite superior/inferior establecido en valores numéricos, fechas y horas), pueden ofrecer adicionalmente validación de formulario nativa; otra tarea tediosa, que, hasta ahora, requería que los desarrolladores crearan resmas de JavaScript o utilizaran algún script/biblioteca de validación ya existente.

Nota: para que los controles de formulario sean validados, necesitan tener un atributo `name`, ya que sin el no serán enviados como parte del formulario.

### required

Uno de los aspectos más comunes de la validación de formularios es la implementación de campos obligatorios, no permitiendo que un formulario sea enviado hasta que ciertas piezas de información

hayan sido introducidas. Esto puede hacerse ahora simplemente añadiéndole el atributo `required` a un elemento `input`, `select` o `textarea`.

```
<input type="text" ... required>
```

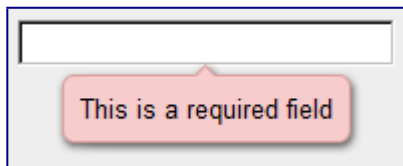


Figura 10: La validación en lado cliente de Opera en acción, mostrando un error para un campo requerido que ha sido dejado vacío.

## type y pattern

Como hemos visto, los desarrolladores ahora pueden especificar los tipos de entradas que esperan en sus campos de formulario. En lugar de simplemente definir campos de entrada de texto, los desarrolladores pueden crear de forma explícita campos de entrada para cosas como números, direcciones de correo electrónico y URLs. Como parte de su validación en el lado cliente, los navegadores pueden ahora comprobar que los datos introducidos por el usuario en estos campos más específicos coinciden con la estructura prevista. En esencia, los navegadores evalúan los valores de la entrada en base a un patrón integrado que define como deben ser las entradas válidas en esos tipos de entrada y le advertirá al usuario cuando su entrada no coincida con los criterios.

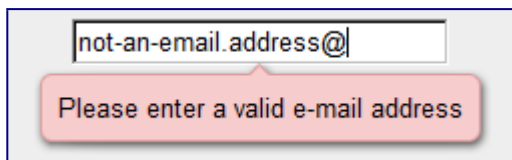


Figura 11: Mensaje de error de Opera para direcciones de correo-e inválidas en un campo de entrada *email*.

Para otros campos de entrada de texto que, no obstante, deban seguir una cierta estructura (por ejemplo, formularios de inicio de sesión donde los nombres de usuario sólo puedan contener una secuencia específica de letras minúsculas y números), los desarrolladores pueden utilizar el atributo `pattern` para especificar su propia expresión regular personalizada.

```
<input type="text" ... pattern="[a-z]{3}[0-9]{3}">
```

## Más ejemplos de Patrones

Patrón	Descripción	Ejemplo		No validaría
<code>^</code>	Coincidencia al principio de la cadena	<code>^Esto</code>	Esto es una cadena	Y esto de aquí
<code>\$</code>	Coincidencia al final de la cadena	<code>final\$</code>	Llegamos al final	
<code>*</code>	El carácter que precede puede aparecer 0, 1 o más veces	<code>ma*</code>	maaaaa o ma o m	
<code>+</code>	El carácter que lo precede debe aparecer por lo menos una vez (1 o más veces)	<code>ma+</code>	maaaaa o ma	m

?	El carácter que lo precede puede aparecer como mucho una vez (0 o 1 vez)	ma?	ma o m	maaaaa
[...]	Cualquier carácter dentro de los paréntesis	a[px]e	ape o axe	ale
[ - ]	Indica un rango de caracteres	[c-k]a	ca o ea	pa
[^...]	Cualquier carácter menos los que están entre paréntesis	a[^px]e	ale o a1e	ape ou Axe
[^ - ]	Cualquier carácter menos los que estén en ese rango	[^c-k]a	pa	ea
{n}	El carácter que lo precede debe aparecer exactamente n veces	bua{2}	buaa	bua
{n,}	El carácter que lo precede debe aparecer n veces o más	bu{2,}	buuuuu o buu	bu
{n,m}	El carácter que lo precede debe aparecer como mínimo n y como máximo m veces	[0-9]{2,4}	32 o 4444	8 o 78978
(...)	Podemos agrupar con paréntesis	(co){2}liso	cocoliso	Coliso
	Una barra vertical separa alternativas	M[0-9]   MP	MP o M9	MA 9M
\d	Cualquier dígito ( es equivalente a [0-9])	\d{2}	23	S7
\D	Cualquier no dígito	\D{2}M	L_M o ABM	4CM
\w	Cualquier letra o dígito	ala\w	ala4 o alag	ala_
\W	Cualquier carácter diferente de letra o dígito	M\Wa	M-a	M1a
.	Cualquier carácter salvo salto de línea			
\n	Salto de línea			
\s	Carácter en blanco, tabulador o salto de línea			