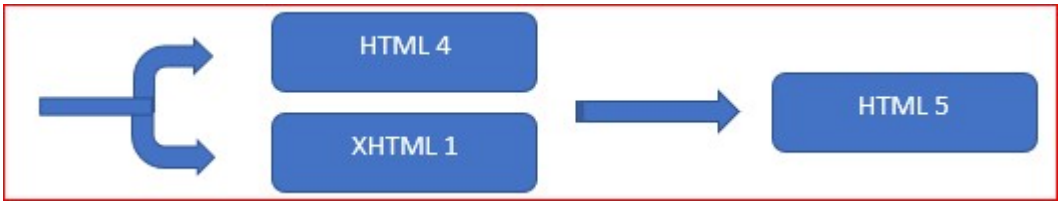
Esta versión de HTML aporta nuevos elementos enfocados a distintas tareas con la presentación, el diseño de la página, facilitar la inclusión de audio y video, mejorar los formularios, almacenar los datos de sesión (evitando usar cookies), permitir la geolocalización del sitio web, construir una superficie de dibujo llamada "*canvas*", arrastrar objetos de un lugar a otro de la página, entre otras.

HTML5 también pretende convertirse en una plataforma de desarrollo agrupando las nuevas tecnologías de desarrollo de aplicaciones web: HTML5, CSS3 y nuevas capacidades de JavaScript, combinados con diferentes formatos de audio y video, animaciones… HTML5 incorpora una biblioteca de funciones lo bastante amplia para que todos los navegadores se comporten de igual forma.

La versión anterior y más usada de HTML, HTML4, carece de las características necesarias para la creación de aplicaciones modernas basadas en un navegador. El uso fuerte de JavaScript ha ayudado a mejorar esto. Hasta ahora, Flash en especial ha sido usado para desarrollar webs que superaran las habilidades de un navegador: audio, video, webcams, micrófonos, datos binarios, animaciones vectoriales, componentes de interfaz complejos, entre muchas otras cosas. Ahora HTML5 es capaz de hacer esto sin necesidad de plugins y con una gran compatibilidad entre navegadores.

En cuanto a la sintaxis, es muy flexible y pone fin a la bifurcación entre HTML y XHTML, convirtiéndose en un estándar para los dos lenguajes.



DOCTYPE

HTML4 y HTML5 son 100% compatibles entre sí. Todo código en HTML 4 seguirá funcionando sin problemas en HTML 5. Para empezar a usar HTML 5 lo único que hay que hacer es colocar este DOCTYPE antes de la etiqueta <html>:

```
<!DOCTYPE html>
```

Es un DOCTYPE mucho más simplificado y que permite usar todas las habilidades de HTML 5 sin que nada de lo que ya tienes programado deje de funcionar.

La declaración de codificación de caracteres (charset) también es muy simple:

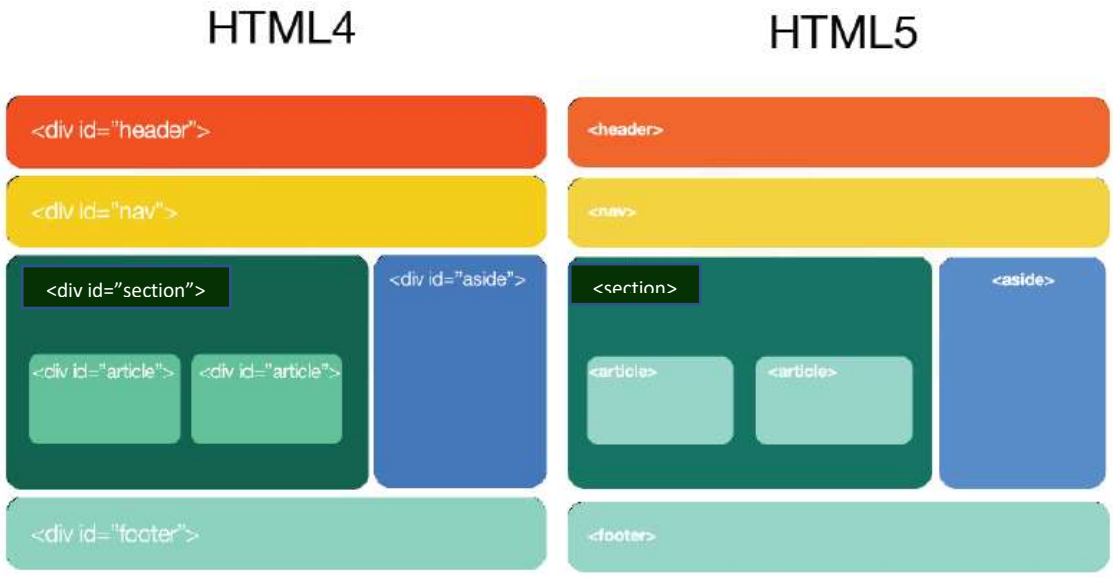
```
<meta charset="UTF-8">
```

Un ejemplo en HTML5 sería:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>Título del documento</title>
  </head>
  <body>
    Contenido del documento ...
  </body>
</html>
```

NUEVA ESTRUCTURA

En HTML4 sólo existía un elemento contenedor sin significado semántico * *.
En HTML5 existen *div*, *section* y *article* que además de cumplir con un comportamiento similar a *div*, aportan semántica.
Para evitar el uso indiscriminado de etiquetas * * como secciones de página, aparecen nuevas etiquetas más específicas que facilitan la comprensión del código, y permitirán a los navegadores y buscadores clasificar dichas secciones según su utilidad e importancia.



Se trata de las etiquetas:

- ** **: es el contenedor genérico, es un elemento a nivel de bloque sin sentido adicional semántico.

- **<header>**: no el elemento head, sino la cabecera de una determinada página. Sirve para insertar información introductoria, como el título, el logo, algún enlace secundario, un cuadro de búsqueda, etc. El elemento header puede estar anidado en otras secciones de la página (es decir, que no solo se utiliza para la cabecera de la página).

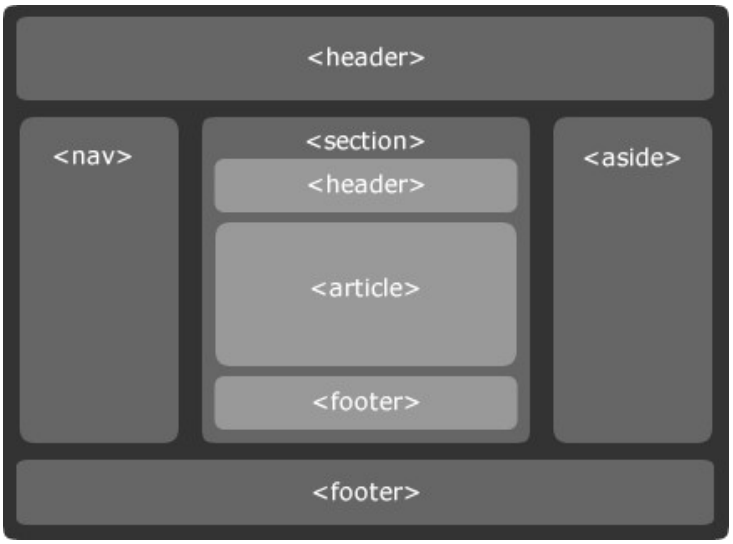
- **<nav>**: sirve para insertar una sección con enlaces a otras páginas o bloques importantes dentro de la página. Se usa para definir el menú o la navegación de la página. No todos los grupos de enlaces en una página deben ser agrupados en un elemento nav, únicamente las secciones que consisten en bloques de navegación más importantes son adecuadas para el elemento de navegación.

- **<section>**: sirve para insertar una sección general dentro del documento. Suele contener los elementos <h1 – h6> para crear una jerarquía de contenidos y puede anidarse para crear subsecciones. Normalmente, tiene un *header* y suele tener un footer. Una sección, en este contexto, es una agrupación temática de los contenidos. Puede ser un capítulo, una sección de un capítulo o básicamente cualquier cosa que incluya su propio encabezado. Una página de inicio de un sitio Web puede ser dividida en secciones para una introducción, noticias, información de contacto etc.

- **<article>**: una entrada independiente en un blog, sitio o revista. Es una parte independiente del documento. También suele llevar un *título* y un *footer.* Cuando se anidan los elementos *article*, los artículos internos están relacionados con el contenido del artículo exterior. Por ejemplo, una entrada de blog en un sitio que acepta comentarios, el elemento *article* principal agrupa el artículo propiamente dicho y otro bloque *article* anidado con los comentarios de los usuarios.

- **<aside>**: sirve para insertar una información relacionada indirectamente con el contenido que tiene a su alrededor, por lo que se considera contenido independiente. Puede utilizarse, por ejemplo, para una barra lateral, publicidad o enlaces de navegación. El elemento *aside* representa una nota, un consejo, una explicación. El elemento puede ser utilizado para efectos de atracción, como las comillas tipográficas o barras laterales, para la publicidad, por grupos de elementos de navegación, y por otro contenido que se considera por separado del contenido principal de la página.

- **<footer>**: en el final como pie de página, donde se incluyen los modos de contacto, autor, copyright, año, mapa del sitio ...



Estructura general:

```
<body>
  <header> … </header>
  <nav> … </nav>
  <section>
    <article> … </article>
    <article> … </article>
    <article> … </article>
    <aside> … </aside>
  </section>
  <footer> … </footer>
</body>
```

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>rty</title>
  </head>
  <body>
    <header>
      <h1>El blog de Carlos</h1>
    </header>
    <nav> Aquí va la botonera de navegación </nav>
    <section>
      <h2>Aquí van los artículos con su título en h2 </h2>
      <article><h2>Aquí va un post, con su título en h2 </h2></article>
      <article><h3>Aquí va un post, con su título en h3 </h3></article>
      <article><h4>Aquí va un post, con su título en h4 </h4></article>
      <aside> Barra lateral en los post </aside>
    </section>
    <aside> Barra lateral con cosas que nadie lee, como cuentas de
          twitter, facebook, posts viejos, etc.
    </aside>
    <footer> Pie de página, copyright, etc. </footer>
  </body>
</html>
```

A su vez, cada artículo puede reproducir la estructura general, es decir, puede tener un <header>, varios <section> y un <footer>.

En HTML4:

```
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<link rel="stylesheet" href="style.css" type="text/css" media="screen"/>
<script src="code.js" type="text/javascript"></script>
```

En HTML5:

```
<metacharset="utf-8">
<link rel="stylesheet" href="style.css"/>
<script src="code.js"> </script>
```

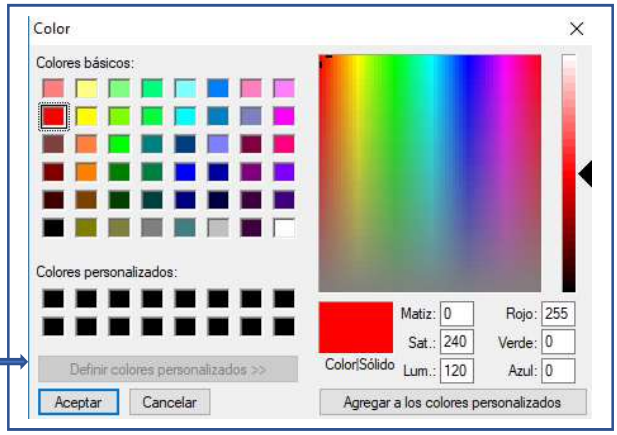
FORMULARIOS

El elemento *<input>* incorpora nuevos "*type*" y nuevos atributos que facilitan la validación de los elementos de entrada.

- **color**

```
<form>
  Selecciona tu color favorito:
  <input type="color" name="favcolor">
</form>
```

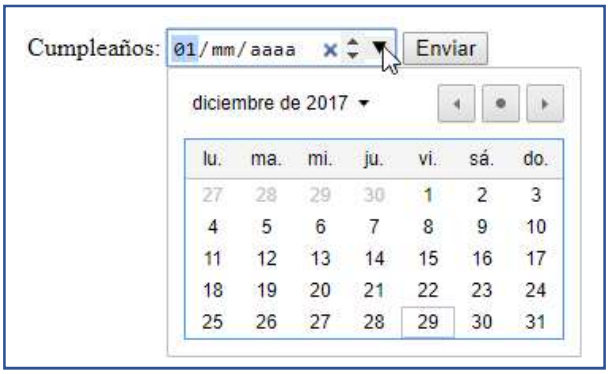
Selecciona tu color favorito: Enviar



- **date**

```
<form>
  Cumpleaños:
  <input type="date" name="bday">
</form>
```

Cumpleaños: dd/mm/aaaa Enviar



Es posible añadir restricciones:

```
<form>
  Fecha anterior al 01-01-1980:<br>
  <input type="date" name="bday" max="1979-12-31"><br>
  Fecha posterior al 01-01-2000:<br>
  <input type="date" name="bday" min="2000-01-02"><br>
</form>
```

Fecha anterior al 01-01-1980:
dd/mm/aaaa

Fecha posterior al 01-01-2000:
01/01/2000

- **datetime-local**

```
<form>
  Cumpleaños (fecha y hora):
  <input type="datetime-local" name="bdaytime">
</form>
```

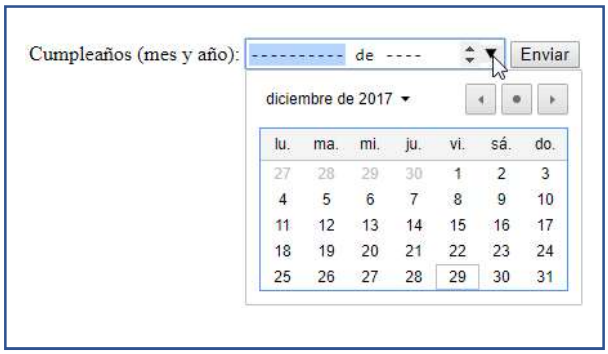
Cumpleaños (fecha y hora): dd/mm/aaaa --:-- Send

- **email**

```
<form>
  E-mail:
  <input type="email" name="email">
</form>
```

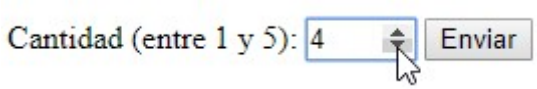
- **month**

```
<form>
  Cumpleaños (mes y año):
  <input type="month" name="bdaymonth">
</form>
```



- **number**

```
<form>
  Cantidad (entre 1 y 5):
  <input type="number" name="quantity" min="1" max="5">
</form>
```



- **range**

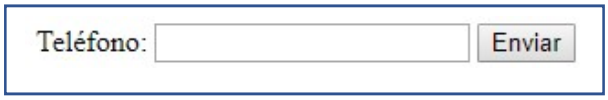
```
<form>
  Puntos:
  <input type="range" name="points" min="0" max="10">
</form>
```

- **search**

```
<form>
  Buscador:
  <input type="search" name="googlesearch">
</form>
```

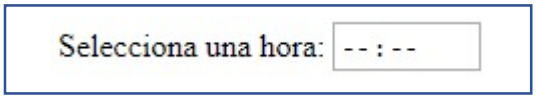
- **tel**

```
<form>
  Teléfono:
  <input type="tel" name="usrtel">
</form>
```



- **time**

```
<form>
  Selecciona una hora:
  <input type="time" name="usr_time">
</form>
```



- **url**

```
<form>
  Página personal:
  <input type="url" name="homepage">
</form>
```

- **week**

```
<form>
  Selecciona una semana:
  <input type="week" name="week_year">
</form>
```

OTROS ELEMENTOS HTML 5

- **mark**

Mark se utiliza para dar importancia a un determinado texto.

```
<p><strong>Note:</strong> La <mark>etiqueta mark</mark> es utilizada para llamar la atención.</p>
```

Nota: La etiqueta mark es utilizada para llamar la atención.

- **progress**

Gracias a la etiqueta <*progress*> podremos incluir **el progreso de una tarea cualquiera que se esté ejecutando**.

```
<progress value="22" max="100"></progress>
```

Descargando:

- **meter**

Gráfico que muestra un progreso.

```
<p>Mostrar un medidor:</p>
<meter value="2" min="0" max="10">2 out of 10</meter><br>
<meter value="0.6">60%</meter>
```

Mostrar un medidor:

- **details**

El elemento *details* permite que un conjunto de información se oculte al usuario inicialmente, pero puede expandirla seleccionando la información definida por el elemento *summary.*

```
<details>
  <summary>Algunos de mis grupos favoritos:</summary>
  <p> - Pereza</p>
  <p> - Estopa</p>
</details>
```

▶ Algunos de mis grupos favoritos:

▼ Algunos de mis grupos favoritos:

- Pereza

- Estopa

- **time**

El elemento time ofrece al navegador información sobre las fechas utilizadas en el documento web. Esta información es de enorme utilidad para los buscadores. Podemos distinguir tres modos de usar esta etiqueta:

```
<p>Abrimos a las <time>10:00</time> cada mañana.</p>
<p>El examen es <time datetime="2008-02-14 20:00">el lunes</time>.</p>
<p>La nave despegará a las <time datetime="2012-11-26T05:00-
07:00">5am del lunes</time>.</p>
```

Abrimos a las 10:00 cada mañana.

El examen es el lunes.

La nave despegará a las 5am del lunes.

ATRIBUTOS

- **value**

Especifica el valor inicial para un campo de entrada.

```
<input type="text" name="firstname" value="Carlos">
```

- **readonly**

Especifica que el campo de entrada es de solo lectura (no se puede cambiar).

```
<input type="text" name="firstname" value="Carlos" readonly>
```

- **disabled**

Especifica que el campo de entrada está deshabilitado. Un campo de entrada deshabilitado no se puede usar, no se puede hacer clic y su valor no se enviará al enviar el formulario.

```
<input type="text" name="firstname" value="Carlos" disabled>
```

- **size**

Especifica el tamaño (en caracteres) para el campo de entrada.

```
<input type="text" name="firstname" value="Carlos" size="40">
```

- **maxlength**

Especifica la longitud máxima permitida para el campo de entrada.

```
<input type="text" name="firstname" value="Carlos" maxlength="40">
```

NUEVOS ATRIBUTOS HTML5

- **Autofocus**

El atributo de autofoco asigna el foco (cursor de escritura) al campo indicado en cuando la página se ha cargado. Solo se puede asignar a un elemento de la página.

```
Nombre:<input type="text" name="fname" autofocus>
```

- **form**

Especifica uno o más formularios a los que pertenece un elemento <input>. Consejo: Para hacer referencia a más de un formulario, usar una lista de identificadores de formulario separados por espacios.

```
<form action="/action_page.php" id="form1">
  Nombre: <input type="text" name="fname"><br>
  <input type="submit" value="Submit">
</form>

Apellidos: <input type="text" name="lname" form="form1">
```

- **formaction**

Especifica la URL de un archivo que procesa el control de entrada cuando se envía el formulario.
El atributo *formaction* anula el atributo de *action* del elemento <*form*>.

```
<form action="/action_page.php">
  Nombre: <input type="text" name="fname"><br>
  Apellidos: <input type="text" name="lname"><br>
  <input type="submit" value="Submit"><br>
  <input type="submit" formaction="/action_page2.php" value="Submit a otra página">
</form>
```

- **formenctype**

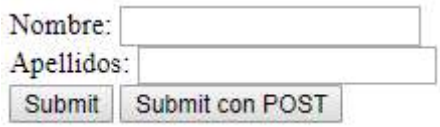
Especifica cómo deben codificarse los datos del formulario cuando se envían (solo para los formularios con method = "post"). El atributo *formenctype* anula el atributo *enctype* del elemento <*form*>.

```
<form action="/action_page_binary.asp" method="post">
  Nombre: <input type="text" name="fname"><br>
  <input type="submit" value="Submit">
  <input type="submit" formenctype="multipart/form-data"
  value="Submit as Multipart/form-data">
</form>
```

- **formmethod**

El atributo *formmethod* define el método HTTP para enviar datos de formulario a la URL de acción. El atributo formmethod anula el atributo de método del elemento <form>.

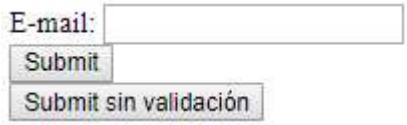
```
<form action="/action_page.php" method="get">
  Nombre: <input type="text" name="fname"><br>
  Apellidos: <input type="text" name="lname"><br>
  <input type="submit" value="Submit">
  <input type="submit" formmethod="post" value="Submit using POST">
</form>
```



- **formnovalidate**

El atributo *formnovalidate* anula el atributo novalidate del elemento <form>.

```
<form action="/action_page.php">
  E-mail: <input type="email" name="userid"><br>
  <input type="submit" value="Submit"><br>
  <input type="submit" formnovalidate value="Submit sin validación">
</form>
```



- **formtarget**

El atributo *formtarget* especifica un nombre o una palabra clave que indica dónde mostrar la respuesta que se recibe después de enviar el formulario. El atributo *formtarget* anula el atributo de destino del elemento <form>.

```
<form action="/action_page.php">
  Nombre: <input type="text" name="fname"><br>
  Apellidos: <input type="text" name="lname"><br>
  <input type="submit" value="Submit normal">
  <input type="submit" formtarget="_blank"
  value="Submit en otra ventana">
</form>
```

- **height and width**

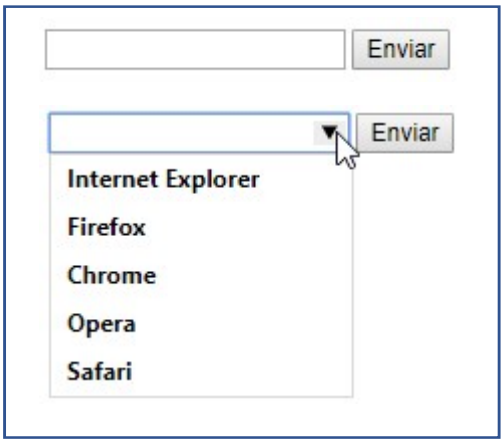
Especifica la altura y el ancho de un elemento <input type = "image">.

```
<input type="image" src="img_submit.gif" alt="Submit" width="48" height="48">
```

- **list**

Usando el atributo *list* con un elemento <*input*> podemos especificar una lista de opciones. Esto permite al usuario seleccionar un valor de la lista o escribir uno que no esté en ella (este tipo de elemento se suele llamar *Combo Boxes*). Los elementos de la lista se deben de indicar utilizando otro nuevo elemento de HTML5, el ***<datalist>***. El cual simplemente nos permite crear una lista de valores.

```
<input list="browsers">

<datalist id="browsers">
  <option value="Internet Explorer">
  <option value="Firefox">
  <option value="Chrome">
  <option value="Opera">
  <option value="Safari">
</datalist>
```



- **min and max**

Especifican los valores mínimo y máximo para un elemento <*input*>. Funcionan con los siguientes tipos de entrada: *number*, *range*, *date*, *datetime-local*, *month*, *time* y *week*.

Fecha anterior a 01-01-1980: <input type="date" name="bday" max="1979-12-31">

Fecha posterior a 01-01-2000: <input type="date" name="bday" min="2000-01-02">

Cantidad (entre 1 y 5): <input type="number" name="cantidad" min="1" max="5">

- **multiple**

Especifica que el usuario puede ingresar más de un valor en el elemento <*input*>. Funciona con los siguientes tipos de entrada: *email* y *file*.

Selecciona imágenes: <input type="file" name="img" multiple>

- **pattern (regexp)**

Este atributo se utiliza para validar la entrada del usuario mediante expresiones regulares. En la dirección "https://es.wikipedia.org/wiki/Expresi%C3%B3n_regular" podemos obtener más información sobre las expresiones regulares. Ejemplo de uso (en Firefox y Chrome funciona):

```
<label for="cp">Código Postal</label>
<input id="cp" name="cp" pattern="[\d]{5}(-[\d]{4})" />
```

- **placeholder**

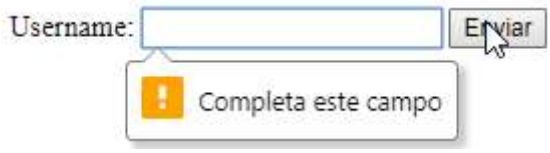
El atributo *placeholder="texto"* se utiliza para colocar el valor de su texto dentro del campo a modo de ayuda. Si se focaliza dicho campo, se elimina el *placeholder*. Si abandonamos el campo sin añadir ningún valor, se vuelve a añadir el *placeholder*. Esta funcionalidad siempre ha requerido del uso de JavaScript para ser llevado a cabo, pero con la nueva especificación este comportamiento puede definirse de la forma:

<input type="text" name="fname" placeholder="nombre">

- **required**

Una de las tareas de validación más extendidas es la de los campos requeridos. La nueva especificación de HTML5 incluye el atributo *required* que nos sirve para definir si un campo es requerido o no. Si un campo requerido está en blanco el formulario no será enviado y además avisará con un mensaje:

Usuario: <input type="text" name="usrname" required>



NOTA: Es un error grave de seguridad validar los formularios únicamente desde el lado del cliente, es imprescindible además realizar la validación en el servidor.

- **step**

Especifica los intervalos numéricos para un elemento <input>.

Ejemplo: si paso = "3", los números permitidos podrían ser -3, 0, 3, 6, etc.

Consejo: El atributo step se puede usar junto con los atributos max y min para crear un rango de valores permitidos.

Funciona con los siguientes tipos de entrada: *number*, *range*, *date*, *datetime-local*, *month*, *time* y *week*.

<input type="number" name="points" step="3">

y los siguientes atributos para <form>:

- **autocomplete**

La mayoría de los navegadores incorporan la funcionalidad de autocompletar algunos campos de los formularios con valores introducidos anteriormente. Esta funcionalidad no siempre resulta útil, sobre todo si alguien nos roba nuestro portátil o dispositivo móvil. La nueva especificación de HTML5 nos permite desactivar el autocompletado en un formulario completo o solo en campos específicos. El atributo *autocomplete* nos permite definir dos valores: "*on*" u "*off*".

<form action="/action_page.php" autocomplete="off">
...
</form>

El código anterior desactivaría el autocompletado de todo el formulario. Si por el contrario solo queremos desactivar el autocompletado de un solo campo podemos especificarlo así:

E-mail: <input type="email" name="email" autocomplete="off">

Esta funcionalidad no se puede emular mediante código JavaScript.

- **novalidate**

Cuando está presente, *novalidate* especifica que los datos del formulario no se deben validar cuando se envíen.

```
<form action="/action_page.php" novalidate>
  E-mail: <input type="email" name="user_email">
  <input type="submit">
</form>
```

INPUT RESTRICCIONES

Attribute	Description
disabled	Specifies that an input field should be disabled
max	Specifies the maximum value for an input field
maxlength	Specifies the maximum number of character for an input field
min	Specifies the minimum value for an input field
pattern	Specifies a regular expression to check the input value against
readonly	Specifies that an input field is read only (cannot be changed)
required	Specifies that an input field is required (must be filled out)
size	Specifies the width (in characters) of an input field
step	Specifies the legal number intervals for an input field
value	Specifies the default value for an input field

ELEMENTOS ELIMINADOS EN HTML 5

Los siguientes elementos HTML 4 se han eliminado en HTML 5:

Removed Element	Use Instead
<acronym>	<abbr>
<applet>	<object>
<basefont>	CSS
<big>	CSS
	CSS
<dir>	
<font>	CSS
<frame>	
<frameset>	
<noframes>	
<strike>	CSS, <s>, or <del>
<tt>	CSS

OBJETOS MULTIMEDIA

HTML 5 incluye soporte nativo multimedia, es decir, permite reproducir videos, animaciones, sonido … sin necesidad de instalar plugins de ninguna clase.

<video>

El elemento *<video>* permite mostrar un video sin la necesidad de plugin (Flash). Las propiedades más importantes son:

- *src*: dirección donde se almacena el video.
- *controls*: se visualiza el panel de control del video: botón de inicio, barra de avance del video etc.
- *autoplay*: el video se inicia inmediatamente luego que la página se carga en el navegador.
- *width*: ancho en píxeles del video.
- *height*: alto en píxeles del video.
- *poster*: sirve para visualizar una imagen si no está disponible el video.
- *loop*: es un booleano que indica la repetición automática del video.

No hay un formato de video universalmente adoptado por todos los navegadores y cada uno soporta *codecs* diferentes. El elemento *video* nos permite recodificar un video en múltiples *códecs*:

```
<video width="320" height="240" controls>
  <source src="movie.mp4" type="video/mp4">
  <source src="movie.ogg" type="video/ogg">
  <source src="movie.mkv" type="video/x-matroska">
Tu navegador no soporta el elemento de video.
</video>
```

<audio>

El objetivo de esta etiqueta es permitir la carga y ejecución de archivos de audio sin requerir un plug-in de Flash o Java. El comité de estandarización W3C deja abierto a cada empresa que desarrolla navegadores los formatos que quieran soportar (así tenemos que algunos soportan mp3, wav, ogg, au). Las propiedades más importantes son:

- *src*: la URL donde se almacena el archivo de audio. Si no definimos la URL la busca en el mismo directorio donde se almacena la página.
- *autoplay*: en caso de estar presente el archivo se ejecuta automáticamente después de cargarse la página sin requerir la intervención del visitante.
- *loop*: el archivo de audio se ejecuta una y otra vez.
- *controls*: indica que se deben mostrar la interface visual del control.
- *autobuffer*: En caso de estar presente indica que primero debe descargarse el archivo en el cliente antes de comenzar a ejecutarse.

Como no hay un formato de audio universalmente adoptado por todos los navegadores el elemento audio permite agregar distintas fuentes:

```
<audio controls autoplay loop>
  <source src="horse.ogg" type="audio/ogg">
  <source src="horse.mp3" type="audio/mpeg">
Tu navegador no soporta el elemento de audio.
</audio>
```

El elemento *source* indica a través de la propiedad *src* la ubicación del archivo de audio respectivo. El orden que disponemos estas fuentes es importante. Primero el navegador busca la primera fuente y verifica que puede reproducir dicho archivo, en caso negativo pasa a la siguiente fuente.

<canvas>

El elemento canvas puede definirse como un entorno para crear imágenes dinámicas. Utilizando su API en JavaScript podemos manipular el elemento canvas para dibujar en él y crear gráficos dinámicos de todo tipo (incluidas interfaces de aplicaciones web completas).
Para empezar a usarlo lo único que hay que especificar son sus dimensiones. El texto que escribamos entre la apertura y cierre de la etiqueta *canvas* solamente será interpretado por navegadores que no soporten esta etiqueta:

```
<canvas id="myCanvas" width="360" height="240">
  <p>Tu navegador no soporta canvas</p>
</canvas>
```

El resto de trabajo con canvas se ha de realizar con código JavaScript. Primero debemos referenciar este elemento y adquirir su contexto:

```
var canvas = document.getElementById('myCanvas');
var context = canvas.getContext('2d');
```

Una vez adquirimos el contexto podemos empezar a dibujar. La API bidimensional ofrece muchas de las herramientas que podemos encontrar en cualquier aplicación de diseño gráfico: trazos, rellenos, gradientes, sombras, formas y curvas Bézier. Los principales métodos disponibles son:

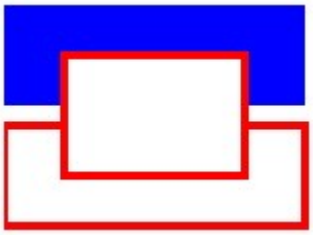
• **fillRect(x, y, width, height)**: dibuja un rectángulo relleno de color según el estilo activado.
• **strokeRect(x, y, width, height)**: dibuja solo el borde de un rectángulo, el interior será transparente.
• **clearRect(x, y, width, height)**: borra el área indicada.
• **beginPath()**: inicializa el dibujado de un "trazo".
• **closePath()**: cierra la figura creando una línea desde el último punto hasta el primero.
• **moveTo(x, y)**: mueve el puntero del trazo hasta las coordenadas indicadas (para poder seguir dibujando).
• **lineTo(x, y)**: dibuja un trazo desde la posición actual hasta las coordenadas indicadas.
• **stroke()**: dibuja el trazo indicado desde el último "beginPath()".
• **fill()**: cierra el trazo definido desde el último "beginPath()" y lo rellena.
• **arc(x, y, radius, startAngle, endAngle, anticlockwise)**: dibuja un arco con centro en "x, y" y el radio definido. Los ángulos se definen en radianes (radianes = (PI/180)*grados) y el último parámetro es un valor booleano.
• **quadraticCurveTo(controlx, controly, x, y)**: dibuja una curva de bezier cuadrática.
• **bezierCurveTo(control1x, control1y, control2x, control2y, x, y)**: dibuja una curva de bezier cúbica.
• **drawImage(x, y)**: dibuja una imagen (como objeto JavaScript) en el canvas.
• **createImageData(width, height)**: crea un objeto ImageData como un array de píxeles para ser manipulado como un array de enteros.
• **getImageData(x, y, w, h)**: carga un objeto ImageData a partir del dibujo actual para ser manipulado.

• **putImageData(imageData, x, y)**: mapea los valores de un objeto ImageData en el dibujo actual.
• **strokeText(string, x, y)**: dibuja una cadena de texto usando solo su borde.
• **fillText(string, x, y)**: dibuja una cadena de texto.

Un ejemplo de dibujado en un objeto *canvas* una vez capturado su contexto:

```
<!DOCTYPE HTML>
<html>
  <head>
    <meta charset="UTF-8">
    <title>canvas</title>
  </head>
  <body>
    <canvas id="myCanvas" width="300" height="300">
      Su navegador no permite utilizar canvas.
    </canvas>

    <script type="text/javascript">

      var canvas = document.getElementById('myCanvas');
      var context = canvas.getContext('2d');

      // Primero definimos las propiedades con las que vamos a dibujar
      context.fillStyle = '#0000ff'; // color de relleno azul
      context.strokeStyle = '#ff0000'; // color de borde rojo
      context.lineWidth = 4; // grosor de línea
      // Y a continuación dibujar algunas figuras
      // rectángulo relleno
      context.fillRect (0, 0, 150, 50);
      // rectángulo solo borde
      context.strokeRect(0, 60, 150, 50);
      // borrar área del canvas
      context.clearRect (30, 25, 90, 60);
      // Orden de coordenadas: izqda,arriba, ancho, largo
      context.strokeRect(30, 25, 90, 60);

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

Obteniendo finalmente un resultado similar a:

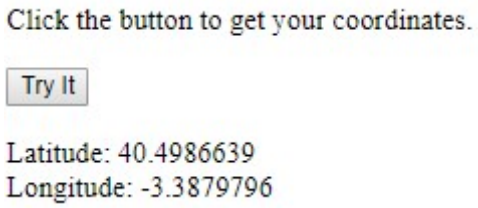


Webs muy importantes están cambiando sus contenidos a canvas y dejando de usar Flash, como Slideshare.

GEOLOCALIZACIÓN HTML5

La geolocalización es la forma de obtener tu posición en el mundo y si quieres, compartir esta información. Existen muchas maneras de descubrir donde te encuentras, por tu dirección IP, la conexión de red inalámbrica, la torre de telefonía móvil por la que se conecta tu móvil, o usando directamente el posicionador GPS.
HTML5 incorpora una nueva funcionalidad para facilitar esta tarea, que dependerá de que el navegador le de soporte.

```
<!DOCTYPE html>
<html>
 <body>
   <p>Click the button to get your coordinates.</p>
   <button onclick="getLocation()">Try It</button>
   <p id="demo"></p>
   <script>
    var x = document.getElementById("demo");
    function getLocation() {
      if (navigator.geolocation) {
        navigator.geolocation.getCurrentPosition(showPosition);
      } else {
        x.innerHTML = "Geolocation is not supported by this browser.";
      }
    }
    function showPosition(position) {
      x.innerHTML = "Latitude: " + position.coords.latitude + "<br>Longitude: " +
position.coords.longitude;
    }
   </script>
 </body>
</html>
```



DRAG AND DROP HTML5

Es una característica que permite arrastrar elementos de un lado a otro.
En HTML5 aparecen:

Nuevos eventos

• **dragstart**: Comienza el arrastrado. El "*target*" del evento será el elemento que está siendo arrastrado.
• **drag**: El elemento se ha desplazado. El "*target*" del evento será el elemento desplazado.
• **dragenter**: Se activa al entrar un elemento que se está arrastrando, dentro de un contenedor. El "*target*" del evento será el elemento contenedor.
• **dragleave**: El elemento arrastrado ha salido del contenedor. El "*target*" del evento será el elemento contenedor.
•**dragover**: El elemento ha sido movido dentro del contenedor. El "*target*" será el contenedor. Como el comportamiento por defecto es cancelar "*drops*", la función debe devolver false o llamar a *preventDefault* para indicar que se puede soltar dentro de ese contenedor.
• **drop**: El elemento arrastrado ha sido soltado en un contenedor. El "*target*" del elemento será el contenedor.
• **dragend**: Se ha dejado de arrastrar el elemento, se haya dejado en un contenedor o no. El "*target*" del evento es el elemento arrastrado.

Para utilizar Drag & Drop

• Definir un objeto como "arrastrable", estableciendo su atributo *draggable="true"* (por defecto "*true*" en imágenes).
• Definir el comportamiento adecuado cuando se detecta un evento relacionado con Drag & Drop.
• Posibilidad de establecer la imagen "*ghost*" mostrada mientras se desplaza.
• Efectos asociados a copiar, mover…

```
<!DOCTYPE HTML>
<html>
<head>
<script>
function allowDrop(ev) {
    ev.preventDefault();
}

function drag(ev) {
    ev.dataTransfer.setData("text", ev.target.id);
}

function drop(ev) {
    ev.preventDefault();
    var data = ev.dataTransfer.getData("text");
    ev.target.appendChild(document.getElementById(data));
}
```

```
</script>
</head>
<body>
<div id="div1" ondrop="drop(event)" ondragover="allowDrop(event)"></div>
<img id="drag1" src="img_logo.gif" draggable="true"
ondragstart="drag(event)" width="336" height="69">
</body>
</html>
```

Drag the W3Schools image into the rectangle:

Drag the W3Schools image into the rectangle: