HTML 5

(parte 2)

octubre 2018

Prof. Ainhoa Iglesias

ÍNDICE:

HTML 5

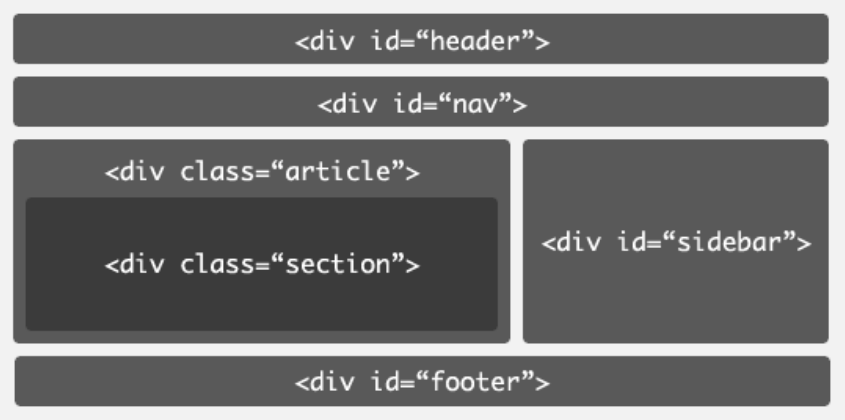
PARTE 1: NOVEDADES en HTML5

1. ELEMENTOS ESTRUCTURALES

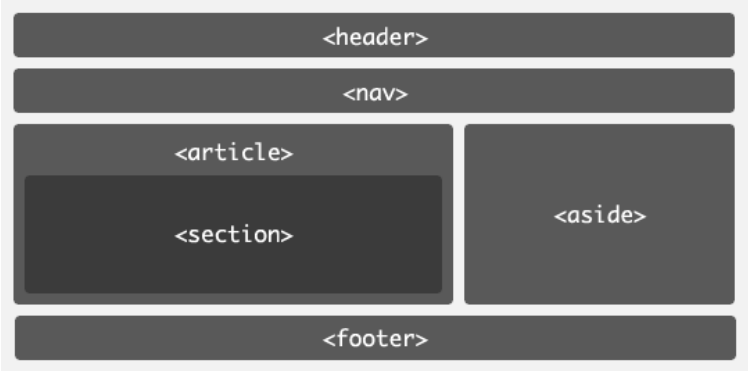
Hasta ahora, utilizábamos el elemento `<div>` para agrupar diversos bloques información y aplicar propiedades CSS. Sin embargo, estos bloques no tenían ningún significado semántico.

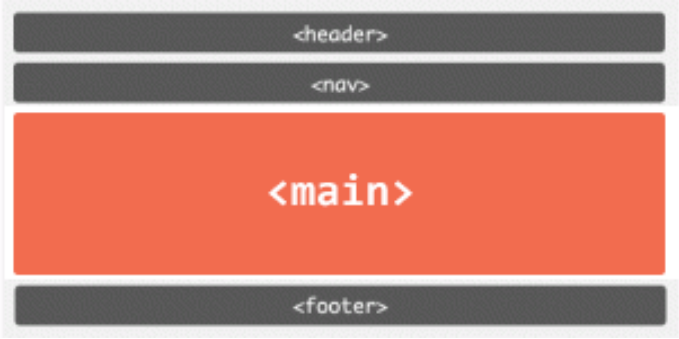
Con los nuevos elementos estructurales de HTML 5, vamos a poder sustituir la mayor parte de los elementos `<div>` y dotar a nuestro documento de una estructura semántica adecuada.

Antes (“divitis”, en muchos casos, aguda):



Ahora* (utilizaremos los elementos estructurales , cuando sea oportuno, para dar sentido y significado a los contenidos):





*** Nota**: como veremos más adelante, un `article` puede contener `section`, y un `section` puede contener `article` (la imagen anterior es un mero ejemplo).

HEADER:

El elemento <header> representa un grupo introductorio de información o también puede usarse para agrupar ayudas de navegación.

En principio está pensado para que se incluya el encabezado de sección (<h1>-<h6>).

También puede usarse para agrupar la tabla de contenidos de una sección, un formulario de búsqueda o logotipos relevantes.

Pueden existir varios <header> en una página (en función del contenido que exista)

Antes:

```
<div id="cabecera">
    <h1>Nombre Empresa</h1>
    [menú idiomas, auxiliar...]
</div>
```

Ahora:

```
<header>
    <h1> Nombre Empresa </h1>
    [menú idiomas, auxiliar...]
</header>
```

FOOTER:

El elemento <footer> representa el pie de la sección de contenido predecesora más cercana, o la sección del contenido raiz.

Normalmente, el elemento <footer> contiene información acerca de su sección, como el autor, enlaces a documentos relacionados, licencias, etc.

También es posible incluir la información que habitualmente se incluye en el elemento <address> (información de contacto del autor o editor, por ejemplo).

El elemento <footer> no tiene por que aparecer en el final de la sección.

Cuando el elemento <footer> contiene secciones enteras, éstas representan apéndices, índices, acuerdos de licencias y otro contenido similar.

Pueden existir varios <footer> en una página (en función del contenido que exista)

Antes:

```
<div id="pie">
    <p> Copyright.. ./ imagen... / texto....</p>
</div>
```

Ahora:

```
<footer>
    <p> Copyright... /  imagen... /  texto....</p>
</footer>
```

NAV:

El elemento `<nav>` representa una sección de la página que contiene enlaces de navegación a zonas de la misma página o a otras páginas.

No todos los grupos de enlaces tienen que implementarse con el elemento `<nav>`. Sólo aquellas secciones que contengan los bloques principales de navegación.

En el caso de los típicos enlaces del "menú auxiliar" (portada, aviso legal, etc…), con el elemento `<footer>` sería suficiente (aunque podemos utilizar además, `<nav>`).

Algunos agentes de usuario (como los lectores de pantalla, por ejemplo), omitirán la lectura de los elementos `<nav>` en una primera lectura (como si hubiéramos implementado un salto de contenido).

Antes:

```
<div id="menu-ppal">
    <ul>
       <li><a href="#">quiénes somos</a></li>
       <li><a href="#">servicios</a></li>
       <li><a href="#">noticias</a></li>
    </ul>
</div>
```

Ahora:

```
<nav id="menu-ppal">
  <ul>
    <li><a href="#">quiénes somos</a></li>
    <li><a href="#">servicios</a></li>
    <li><a href="#">noticias</a></li>
  </ul>
</nav>
```

ASIDE:

El elemento `<aside>` representa una sección de la página cuyo contenido está relacionado tangencialmente con el contenido que tiene a su alrededor, pero se considera contenido independiente, adicional.

Se puede usar para implementar barras laterales, publicidad, elementos con efectos tipográficos (como las citas)

No es adecuado utilizarlo para estructurar contenido explicativo, porque forma parte del contenido principal.

Antes:

```
<div id="publicidad">
    [... diferentes bloques de publicidad...]
</div>
```

Ahora:

```
<aside id="publicidad">
    [... diferentes bloques de publicidad...]
</aside>
```

FIGURE / FIGCAPTION:

El elemento <figure> permite asociar contenido embebido (diagramas, ilustraciones, fotos, video, audio, citas…) con un texto. Debe hacerse una referencia en el texto de la página hacia el bloque `<figure>`.

El elemento <figcaption> se utilizará para implementar el texto asociado al contenido embebido.

Antes:

```
<div id="bloque-foto">
    <img...>
    Pie de foto
</div>
```

Ahora:

```
<figure>
    <img...>
    <figcaption>
       Pie de foto
    </figcaption>
</figure>
```

ARTICLE:

El elemento `<article>` representa una composición autónoma en un documento, página, aplicación o sitio web, con intención de poder ser reutilizado (por ejemplo, en un RSS).

Puede utilizarse en un artículo de un foro, revista, artículo de periódico, entrada (post) de un blog, comentario escrito por un usuario, widget interactivo o gadget e incluso en cualquier otro elemento independiente de contenido.

Cuando existen elementos `<article>` anidados, los elementos `<article>` interiores estarían relacionados con los `<article>` exteriores (p.e un `<article>` que contenga un artículo de un blog, contendrá también `<article>` para los comentarios de un usuario).

Antes:

```
<div class="post">
    <h2>Apúntate al curso de CSS 3 y HTML 5</h2>
    […contenido del post…]
</div>
```

Ahora:

```
<article class="post">
    <h2>Apúntate al curso de CSS 3 y HTML 5</h2>
    […contenido del post…]
</article>
```

SECTION:

El elemento `<section>` representa una sección genérica de un documento o aplicación. Una sección, en este contexto, es una agrupación temática de contenido, habitualmente con un encabezado.

Se recomienda usar `<article>` en vez de `<section>` cuando tenga sentido sindicar los contenidos del elemento.

El elemento `<section>` no es un elemento contenedor genérico. Si sólo se necesita el elemento para aplicar estilos, entonces se deberá utilizar el elemento `<div>`.

Antes:

```
<div id="bloque-noticias">
    [listado de las 5 noticias más recientes]
</div>

<div id="bloque-eventos">
    [listado de los 5 eventos más recientes]
</div>

<div id="bloque-entrevistas">
    [listado de las 5 entrevistas más recientes]
</div>
```

Ahora:

```
<section id="bloque-noticias">
    [listado de las 5 noticias más recientes]
</section>

<section id="bloque-eventos">
    [listado de los 5 eventos más recientes]
</section>

<section id="bloque-entrevistas">
    [listado de las 5 entrevistas más recientes]
</section>
```

MAIN:

El elemento `<main>` representa la parte de la página donde se encuentra el contenido principal de la misma. Debe ser un contenido único en el documento y excluye el contenido que se repite en la web como por ejemplo menús de navegación, información de copyright, logotipo, banners, formulario de búsqueda -salvo que estemos en la página específica de búsqueda-).

No puede haber más de un elemento `<main>` visible en la página (puede haber más en el código, pero se deberán ocultar a la vista)

El elemento `<main>` no puede implementarse como descendiente de los elementos `<article>`, `<aside>`, `<footer>`, `<header>` o `<nav>`.

Ejemplo:

```
<main>

    <h1>Noticias</h1>
    <p>Noticias locales siempre en primicia</p>

    <article>
       <h2>Noticia 1</h2>
       <p>Descripción noticia 1</p>
       </article>

    <article>
       <h2>Noticia 2</h2>
       <p>Descripción noticia 2</p>
    </article>

</main>
```

TIME:

El elemento `<time>` representa una fecha/hora[1] en base al calendario Gregoriano.

(El elemento desapareció[2] de la especificación en octubre de 2011[3] y volvió al mes siguiente[4])

En la especificación HTML 5 se contemplan diferentes maneras[5] de implementar las fechas/horas.

Antes:

```
<div class="post">
    <h2>Apúntate al curso de CSS 3 y HTML 5</h2>
    <p class="fecha">16 de noviembre de 2012</p>
    [...contenido del post...]
</div>
```

Ahora:

```
<article class="post">
    <h2>Apúntate al curso de CSS 3 y HTML 5</h2>
    <p>
       <time datetime="2012-04-26" pubdate>
             26 de abril de 2010
       </time>
    </p>
    [...contenido del post...]
</article>
```

[1] http://www.whatwg.org/specs/web-apps/current-work/multipage/text-level-semantics.html#the-time-element
[2] http://www.iandevlin.com/blog/2011/10/html5/on-the-disappearance-of-html5
[3] http://www.brucelawson.co.uk/2011/goodbye-html5-time-hello-data/
[4] http://www.brucelawson.co.uk/2011/the-return-of-time/
[5] http://www.whatwg.org/specs/web-apps/current-work/multipage/common-microsyntaxes.html#valid-global-date-and-time-string

PICTURE:

El elemento `<picture>`[6] es de reciente creación y tiene como objetivo dar una solución al Responsive Design de las imágenes. Aún está en desarrollo y el soporte es bajo:

```
<picture>
   <source media="(min-width: 1100px)" src="grande.jpg">
   <source media="(min-width: 750px)" src="mediana.jpg">
   <source src="peque.jpg">
   <img src="peque.jpg" alt="">
   <p>Texto Accessible Alternativo</p>
</picture>
```

Y mediante el atributo `srcset` (también de nueva creación), es posible discriminar las imágenes en base a la densidad de píxel (retina):

```
<picture>
   <source media="(min-width: 1100px)" srcset=" grande-1.jpg 1x,
grande-2.jpg 2x">
   <source media="(min-width: 750px)" srcset=" mediana-1.jpg 1x,
mediana-2.jpg 2x">
   <source srcset=" peque-1.jpg 1x, peque-2.jpg 2x">
   <img src=" peque-1.jpg" alt="">
   <p>Texto Accessible Alternativo</p>
</picture>
```

EJEMPLO GLOBAL:

Página en HTML 4[7] vs. Página en HTML 5[8]

[6] http://w3c.github.io/html/semantics-embedded-content.html#the-picture-element
[7] http://diveintohtml5.info/examples/blog-original.html
[8] http://diveintohtml5.info/examples/blog-html5.html

2. OTROS ELEMENTOS

Algunos elementos son nuevos y otros se han reutilizado, dejando atrás su función como elemento de presentación para convertirse en elementos semánticos[9] [10].

b

El elemento `<b>` representa un fragmento de texto que se resalta por ser importante en relación al contenido: palabras clave de un documento, nombres de productos en una revisión del mismo, o cualquier otro fragmento de texto que se represente en negrita en modo escrito.

```
<h2>Caracoles Lentos</h2>
<p>Los <b class="nom-product">Caracoles Lentos</b> son un nuevo
producto de la empresa alimenticia....</p>
```

i

El elemento `<i>` representa un fragmento de texto en voz alternativa o contenido que se presenta con itálica en modo escrito: palabras en otro idioma (utilizando el atributo `lang`), términos técnicos y taxonómicos, notación musical, pensamientos, cambios en el estado de ánimo, referencia a contenido escrito a mano...

Es recomendable utilizar clases para indicar el significado que se le quiere dar al elemento `<i>` en cada caso (de este modo se demuestra que no se está utilizando como elemento de diseño):

```
<p>Ayer probé un plato de <i lang="ca">escargots</i>, caracoles
de la familia de los <i class="taxonomy">Helix aspersa</i>.</p>
```

small

El elemento `<small>` se utilizará para el contenido llamado comúnmente de "letra pequeña" (avisos legales y similares) y para comentarios adicionales a un texto (no confundirlo con el elemento `<aside>`).

En definitiva, para aquellos contenidos que en modo escrito se representan con letra más pequeña.

```
<small>
     <a rel="license" href="http://creativecommons.org/licenses/by-sa/3.0/">
        Creative Commons Attribution Share-alike license
     </a>
</small>
```

```
<p> El diseño está basado en la plantilla Keko de Mkels
     <small>(http://www.mkels.com/demo/)</small>
</p>
```

[9] http://html5doctor.com/i-b-em-strong-element/
[10] http://html5doctor.com/small-hr-element/

Otros elementos

`<mark>`: representa un contenido remarcado[11] que es relevante para el usuario.

`<details>`: marca un texto como información adicional.

`<command>`: representa un comando que el usuario puede ejecutar.

`<ruby>`: permite la inserción de anotaciones para lenguajes asiáticos.

Etc.

Nota: es muy importante no utilizar elementos obsoletos que no forman parte de la especificación HTML 5 (aunque en su momento sí formaron parte), como el elemento <hgroup>[12] que a día de hoy se sigue utilizando indiscriminadamente a pesar de que lleva muchos años eliminado de la especificación (desde 2013).

[11] http://html5doctor.com/draw-attention-with-mark/

[12] http://lists.w3.org/Archives/Public/public-html-admin/2013Apr/0003.html
http://html5doctor.com/the-hgroup-element/

3. FORMULARIOS

HTML 5 aumenta los tipos de campos de formulario[13] y ofrece nuevas funcionalidades[14].

input:

email:

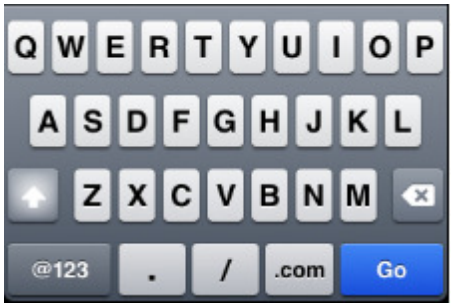
Dirección de e-mail.

```
<input type="email">
```

url:

Dirección URL.

```
<input type="url">
```



tel:

Número telefónico

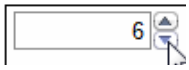
```
<input type="tel">
```

[13] http://diveintohtml5.info/forms.html
[14] http://miketaylr.com/pres/html5/forms2.html

number:

Aumenta o añade un número, mediante botones:

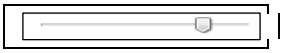
```
<input type="number" min="0" max="6" step="1" value="6">
```



range:

Selecciona un valor dentro del rango especificado:

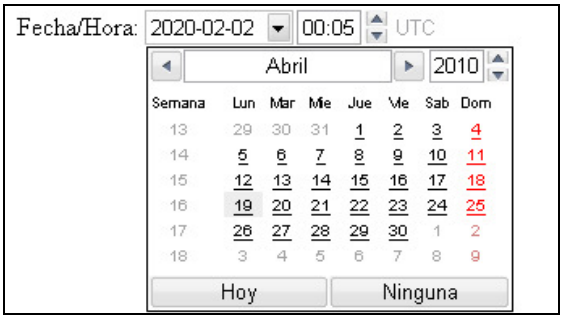
```
<input type="range" min="0" max="10" step="1" value="10">
```



date /time / month / week:

Selección de Fecha, hora, día, mes o semana concreto en un calendario:

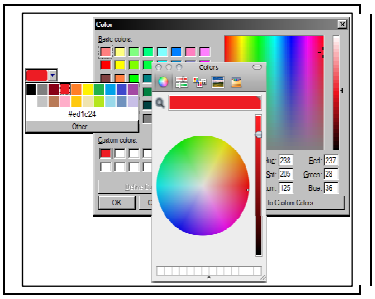
```
<input type="date"> <input type="time">
```



color:

Provee mecanismos para que el usuario inserte un color RGB

```
<input type="color">
```



search:

Campo de buscador.

```
<input type="search">
```

datalist

Se carga una lista de opciones asociadas a un campo input y permite un autocompletado del texto que escribe el usuario, si éste se encuentra en la lista de opciones que ofrecemos[15]:

```
<input list="deportes">
<datalist id="deportes">
    <option value="Baloncesto">
    <option value="Tenis">
    <option value="Motociclismo">
</datalist>
```

También es posible visualizar una etiqueta asociada a cada valor:

```
<input type="text" list="tratamiento">
<datalist id="tratamiento">
    <option label="Sr" value="Señor">
    <option label="Sra" value="Señora">
</datalist>
```

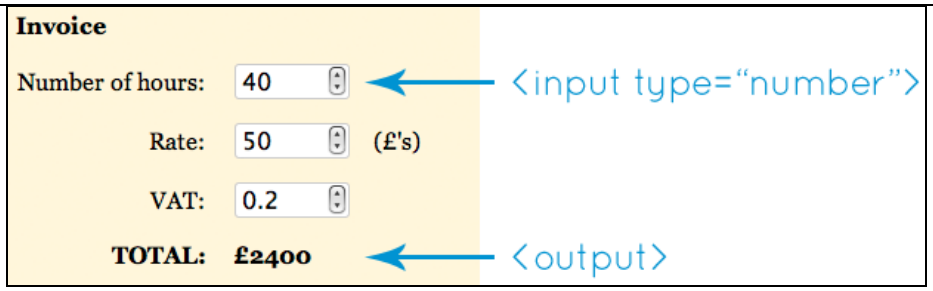
Las opciones pueden guardarse en un archivo xml y enlazar el elemento `<datalist>` a dicho archivo[16].

[15] https://developer.mozilla.org/en-US/docs/Web/HTML/Element/datalist /
[16] http://html5.org/demos/dev.opera.com/article-examples.html

output

El campo `output` se utiliza para representar el resultado de una operación (por ejemplo, una multiplicación de 2 campos, cálculo de edad[17], etc... mediante javascript)

```
<output></output>
```



keygen

Crea una pareja de Clave Pública y Clave Privada[18].

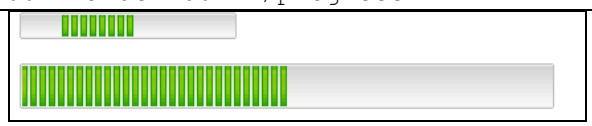
La clave privada se cifra y se almacena en la base de datos local de contraseñas. La clave pública se envía con el formulario.

progress y meter

Estos dos elementos son similares y se utilizan para representar un progreso.

Normalmente `progress`[19] representa una barra de progresos, para indicar el porcentaje completado de una tarea

```
<progress max="100" value="50"></progress>
```



Mientras que `meter`[20] es un indicador genérico.

```
<meter min="0" low="20" optimum="50" high="80" max="100"
value="90"></meter>
```

[17] https://developer.mozilla.org/en-US/docs/Web/HTML/Element/output
[18] http://wufoo.com/html5/elements/4-keygen.html
[19] https://css-tricks.com/html5-progress-element/
[20] https://css-tricks.com/html5-meter-element/

Nuevos Atributos

Min y Max

Determina el valor máximo y mínimo que puede tener un campo (tal y como hemos visto en los `input` de tipo `number, range, progress y meter`):

```
<input name="edad" type="number" min="18" max="25">
```

Step

Indica el incremento entre un valor y el siguiente:

```
<input name="tiempo-consulta" type="number" min="15" max="60"
step="15">
```

Low, Optimus, High

Determina el valor mínimo, óptimo y máximo de un `meter`:

```
<meter min="0" low="20" optimum="50" high="80" max="100"
value="90">
```

placeholder

Permite incluir un texto a modo de ayuda para rellenar un campo de formulario:

```
<input name="termino" placeholder="Buscar en la web">
```

La diferencia entre `placeholder` y el atributo `value` (que utilizábamos anteriormente), es que no existe valor por defecto en el campo y cuando el usuario gane el foco de este campo, el texto desaparece.

El atributo `placeholder` no debe usarse como alternativa al elemento `<label>` Cada uno tiene una misión diferente.

autofocus

Permite forzar al navegador a que sitúe el foco en un campo de formulario determinado, de manera nativa (sin necesidad de utilizar javascript):

```
<input name="termino" autofocus>
```

autocomplete

Se utiliza para activar o desactivar el autocompletado de un formulario o de un campo concreto: [on / off]

```
<form autocomplete="off">
<input type="text" name="nombre" autocomplete="off">
```

spellcheck[21]

Activa o desactiva la propiedad de revisión ortográfica en un contenido editable: `textarea o un input type="text"`: [true / false]

```
<input type="text" spellcheck="false">
<textarea spellcheck="true">
```

multiple:

Para los campos de tipo archivo, será posible seleccionar más de un archivo de una vez[22]:

```
<input type="file" multiple="multiple">
```

Para ofrecer un método alternativo a navegadores que no soporten este atributo, se puede utilizar un script js, como el jQuery Multiple File Upload Plugin [23]

Validación de formularios[24] [25]

Además de las validaciones que proporcionan por sí mismos los campos y atributos vistos hasta ahora, existen 3 atributos extra para la validación de un formulario:

required

Con el objetivo de facilitar la validación del formulario, el atributo `required` obliga a rellenar el campo al que se aplica el atributo.

```
<input type="text" name="usuario" required>
```

Este atributo sólo se puede aplicar a los campos de tipo `text, search, url, telephone, email, password, date pickers, number, checkbox, radio, y file`.

[21] http://blog.whatwg.org/the-road-to-html-5-spellchecking
[22] http://www.wufoo.com/html5/attributes/08-multiple.html
[23] http://www.fyneworks.com/jquery/multiple-file-upload/
[24] http://www.alistapart.com/d/forward-thinking-form-validation/enhanced.html
[25] http://blog.mozilla.com/webdev/2011/03/14/html5-form-validation-on-sumo/

novalidate

Este atributo especifica que un formulario o un campo `input` no se debe validar cuando se ejecute el formulario.

```
<form action=" " novalidate="novalidate">
E-mail: <input type="email" name="user_email">
<input type="submit" value="enviar">
</form>
```

Este atributo sólo se puede aplicar a: `form, text, search, url, telephone, email, password, date pickers, range, y color`.

pattern

Podemos crear un patrón de entrada de datos, basado en expresiones regulares, para un campo determinado, con el fin de que sea el propio navegador quien realice la validación de ese campo, en base al patrón definido, sin necesidad de utilizar una validación javascript[26]:

```
<input type="text" name="codigo-idioma" pattern="[A-z]{2}"
title="Código de idioma con 2 letras">
```

```
<input id="telefono" name="telefono" type="tel"
placeholder="Patron: 1-234-567-8910" required size="50"
pattern="([0-9]{1}(-[0-9]{3})(-[0-9]{3})(-[0-9]{4}))">
```

Este atributo sólo se puede aplicar a los campos de tipo `text, search, url, telephone, email y password`.

accept

El atributo `accept` se utiliza para limitar el formato de los archivos válidos en un `input type="file"`, en base a su MIME type (audio/*, video/*, image/*...)[27]:

```
<input type=file accept="image/*">
```

***Nota**: Podremos personalizar las validaciones mediante la API de Validación (que veremos más adelante)*

[26] http://html5pattern.com/

[27] http://www.wufoo.com/html5/attributes/07-accept.html

4. AUDIO Y VIDEO

Con HTML 5 es posible insertar audio[28] y video[29] de forma nativa [30], sin depender de plugins de ningún tipo (flash, silverlight…) [31]:

Antes:

```
<object classid="clsid:xxxxyyyyyzzzzzz" width="400" height="300"
codebase="http://download.macromedia.com/pub/shockwave/cabs/flash/swflas
h.cab#version=6,0,40,0">
<param name="allowFullScreen" value="true" />
<param name="allowscriptaccess" value="always" />
<param name="src" value="http://www.youtube.com/v/ACCDDCC" />
<param name="allowfullscreen" value="true" />
<embed type="application/x-shockwave-flash" width="400" height="300"
src="http://www.youtube.com/v/ ACCDDCC " allowscriptaccess="always"
allowfullscreen="true">
</embed>
</object>
```

Ahora:

```
<video width="640"  height="360"
src="http://www.youtube.com/v/ACCDDCC.mp4" controls autobuffer>
    <p>
       Alternativa para navegadores sin soporte:
       <a href=" http://www.youtube.com/v/ACCDDCC.mp4"> </a>
    </p>
</video>
<audio src="musica.ogg" controls autobuffer >
    <a href="musica.ogg">Descarga audio</a>
</audio>
```

Atributos:

En el reproductor de audio/video podemos utilizar los siguientes atributos:

src

La URL del contenido de audio/video (puede reemplazarse por `source`).

```
<audio src="cancion.mp3"></audio>
```

autoplay (true|false):

Indica si el archivo debe reproducirse automáticamente o no.

```
<audio src="cancion.mp3" autoplay="false"></audio>
```

loop (true|false):

Indica si el archivo debe volverse a reproducir una vez llegado a su fin.

```
<audio src="cancion.mp3" loop="true"></audio>
```

[28] http://html5doctor.com/native-audio-in-the-browser/
[29] http://html5doctor.com/the-video-element/
[30] http://dev.opera.com/articles/view/everything-you-need-to-know-about-html5-video-and-audio/
[31] http://www.youtube.com/html5

muted (true|false):

Indica si el sonido debe estar activado inicialmente o no. Si no se indica nada, el sonido estará activado.

```
<audio src="cancion.mp3" muted="true"></audio>
```

preload (none|metadata|auto):

Indica al navegador cierta información del archivo.

La diferencia entre `metadata` y `auto` es que `metadata` sólo pre-carga la información del archivo (p.e. la duración que va a tener) y con `auto` se cargará todo el archivo (independientemente de que el usuario lo reproduzca o no).

```
<video src="video.mp4" preload="auto"></video>
```

controls:

Indica si se deben mostrar los controles de reproducción o no.

```
<video src="video.mp4" controls></video>
```

source

Debido al problema de soporte de formatos entre los diferentes navegadores, es recomendable ofrecer el contenido en diferentes formatos. Para indicarle al navegador cuáles son, prescindiremos del atributo `src` (ya que sólo puede enlazar con un único archivo) y lo reemplazaremos por varias instancias del atributo `source` (tantas como formatos dispongamos):

```
<audio controls>
    <source src="cancion.mp3">
    <source src="cancion.ogg">
<!—solución si no hay soporte (enlace descarga, flash...) -->
</audio>
```

poster

Si el video no se carga por alguna razón, podemos establecer una imagen estática (un fotograma del video) que se muestre en ese caso.

```
<video width="250" height="200" src="video.mp4" controls
poster="imagen.png">
</video>
```

Pistas:

El elemento `<track>` permite ofrecer subtítulos, títulos, descripciones, capítulos o metadatos para el elemento multimedia.

La información textual se implementa en un archivo separado del elemento multimedia, que puede estar en diferentes formatos (siempre y cuando sean formatos soportado por los navegadores), aunque es recomendable usar el éstándar que ofrece el W3C, el formato WebVTT.

```
<video controls poster="/images/sample.gif">
    <source src="sample.mp4" type="video/mp4">
    <source src="sample.ogv" type="video/ogv">
    <track kind="captions" src="sampleCaptions.vtt"
srclang="en">
    <track kind="descriptions" src="sampleDescriptions.vtt"
srclang="en">
    <track kind="chapters" src="sampleChapters.vtt"
srclang="en">
    <track kind="subtitles" src="sampleSubtitles_es.vtt"
srclang="de">
    <track kind="subtitles" src="sampleSubtitles_en.vtt"
    <track kind="metadata" src="keyStage1.vtt" srclang="en"
label="Key Stage 1">
    <track kind="metadata" src="keyStage2.vtt" srclang="en"
label="Key Stage 2">
    <track kind="metadata" src="keyStage3.vtt" srclang="en"
label="Key Stage 3">
    <!-- Fallback -->
    ...
</video>
```

Las pistas pueden contener varios atributos:

kind:

Dependiendo del valor del atributo kind, el elemento track puede tener 5 funciones diferentes:

subtitles (valor por defecto): una transcripción o traducción del audio. Los navegadores deberían mostrar esta información como texto superpuesto en el video..

captions: información relevante de audio para ayudar a los usuario que no pueden escuchar apropiadamente el recurso, como cuando el usuario es sordo, cuando el recurso se encuentra silenciado o cuando el ruido del ambiente es elevado. Los navegadores deberían mostrar esta información como texto superpuesto en el video.

descriptions: una descripción que explica al video (o a una parte de él), pensada para ayudar a los usuarios que no pueden verlo apropiadamente, como cuando el usuario es ciego, cuando está conduciendo o cuando el brillo de su pantalla tiene baja intensidad. Los navegadores deberían usar sintetizadores de voz para reproducir el contenido textual.

chapters: títulos de los capítulos, pensados para permitir a los usuarios navegar el recurso de medios. Los navegadores pueden proveerlos como una lista interactiva.

metadata: pensado únicamente para su uso con programas, no son mostrados por el navegador.

srclang:

Indica el idioma de la pista.

label:

Indica el nombre/título de la pista.

default:

Indica la pista que se cargará por defecto

Controles personalizados:

Es posible programar y diseñar nuestros propios controles personalizados para el reproductor de audio/video.

Para ello, será necesario programarlo con javascript, utilizando los atributos y métodos habilitados para tal efecto[32], o bien utilizar alguna librería ya creada.

Accesibilidad en los elementos Audio y Video:

Aunque estemos hablando de un soporte nativo de audio y video por parte de los navegadores, sin ningún plugin externo aparte, los problemas de accesibilidad ligados al propio contenido multimedia siguen estando presentes, además de los problemas derivados de los controles del reproductor.

Además de dar la correspondiente alternativa textual al contenido de audio/video, será necesario que el archivo no se reproduzca automáticamente y los controles puedan ser activados no sólo con el ratón, sino también con teclado o con cualquier otro dispositivo de entrada [33] [34].

Soporte de video:

Inicialmente, la idea era que todos los navegadores soportaran tanto Ogg Theora como H.264, pero:

Ogg es open source y libre de licencia.

H.264 está patentado por MPEG.

Google lanzó WebM[35] en 2010, un nuevo formato[36], con el que empezó otra guerra de formatos[37] [38].

Pero la guerra parece que llegaba a su fin. En marzo de 2012, Mozilla anunció[39] que finalmente iba a dar soporte al formato H.264 en futuras versiones de Firefox. (bajo Windows XP no hay soporte)

Soporte de audio:

En el caso de Audio, nos encontramos hasta 4 formatos diferentes, con un soporte muy pobre por parte de los navegadores.

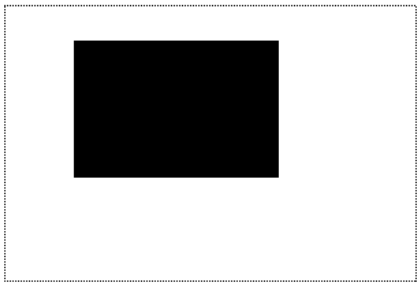
[32] http://www.broken-links.com/2009/10/06/building-html5-video-controls-with-javascript/
[33] http://dev.opera.com/articles/view/accessible-html5-video-with-javascripted-captions/
[34] http://www.brucelawson.co.uk/2009/accessibility-of-html5-video-and-audio-elements/
[35] http://www.webmproject.org/
[36] http://www.anieto2k.com/2010/05/19/webm-el-codec-de-video-open-source-de-google/
[37] http://www.anexom.es/servicios-en-la-red/videos-online/la-guerra-del-video-en-html5-h-264-ogg-theora-y-webm-i/
[38] http://www.anexom.es/servicios-en-la-red/videos-online/la-guerra-del-video-en-html5-h-264-ogg-theora-y-webm-y-ii/
[39] http://blog.lizardwrangler.com/2012/03/18/video-user-experience-and-our-mission/

5. CANVAS

El elemento <canvas>[40] es un mapa de bits que puede usarse para renderizar gráficos, juegos, o cualquier otra imagen visual, en tiempo real.

```
<canvas id="canvas-prueba" width="300" height="200"></canvas>
```

Es un "lienzo en blanco" sobre el que podemos dibujar lo que queramos, e interactuar con él (mediante javascript).



Dibuja!!

```
<script type="text/javascript">
    function dibuja() {
       var b_canvas = document.getElementById("canvas-prueba");
       var b_context = b_canvas.getContext("2d");
       b_context.fillRect(50, 25, 150, 100);
    }
</script>
<p>
    <a href="#" onclick="dibuja();return false">Dibuja!</a>
</p>
```

API 2D

Relleno, borde y líneas básicas:

Relleno: `fillStyle`

Borde: `strokeStyle`

Grosor de las líneas/bordes: `lineWidth`

Rectángulos rellenos: `fillRect`

Rectángulos vacíos (sólo borde): `strokeRect`

Limpiar una parte de canvas: `clearRect`

```
context.fillStyle   = '#abc';
context.strokeStyle = '#000';
```

[40] http://diveintohtml5.info/canvas.html

```
context.lineWidth   = 2;

//x,y,width,height
context.fillRect  (0, 0,  70, 40);
context.strokeRect(50,10, 50, 35);
context.clearRect (30,25, 30, 10);
```

Paths (caminos):

Gracias a la propiedad Paths, podemos dibujar formas personalizadas.

Primero dibujaremos el contorno y estableceremos los atributos de relleno y borde.

Después, declararemos el comienzo del camino con `beginPath()` y procederemos a definirlo. Una vez finalizado, tendremos que aplicar el relleno y borde definido y cerrar el camino con `closePath()`.

```
context.fillStyle   = '#abc';
context.strokeStyle = '#000';
context.lineWidth   = 2;


context.beginPath();
// Coordenadas (x,y)
context.moveTo(10, 10);
context.lineTo(100, 10);
context.lineTo(10, 100);
context.lineTo(10, 10);

context.fill();
context.stroke();
context.closePath();
```

Inserción de imágenes:

Podemos insertar imágenes en el `canvas` mediante la propiedad `drawImage` y varios argumentos:

Imagen básica: fuente de la imagen y coordenadas X,Y para situarla en `canvas`.

Imagen media: los 3 argumentos de la imagen básica, más la anchura y altura de la imagen.

Imagen avanzada: los 5 argumentos anteriores y 4 más: coordenadas X,Y, altura y anchura dentro de la imagen. De este modo, podemos recortar dinámicamente la imagen y mostrar únicamente la porción que deseemos en el `canvas`.

```
// Básica: imagen, coord canvas. x , coord canvas. y).
context.drawImage(img_elem, dx, dy);
```

```
// Media: imagen, coord canvas. x , coord canvas. y, width,
height.
context.drawImage(img_elem, dx, dy, dw, dh);

// Avanzada: imagen, coord. imagen x , coord. imagen y, width
imagen, height imagen, coord canvas. x , coord canvas. y, width,
height.
context.drawImage(img_elem, sx, sy, sw, sh, dx, dy, dw, dh);
```

Puedes consultar más ejemplos / tutoriales [41] [42] [43]

Otras posibilidades:

No hay límites para el elemento canvas, podemos insertar texto[44], pintar[45], manipular elementos, crear gráficos[46] [47], comics, juegos[48]...

Las posibilidades son infinitas!![49] [50] [51] [52] [53]

[41] http://www.html5canvastutorials.com/
[42] https://developer.mozilla.org/es/docs/Web/Guide/HTML/Canvas_tutorial
[43] https://developer.mozilla.org/en-US/docs/Web/API/Canvas_API
[44] http://dev.opera.com/articles/view/html-5-canvas-the-basics/
[45] http://dev.opera.com/articles/view/html5-canvas-painting/
[46] http://d3js.org/
[47] http://www.effectgames.com/demos/canvascycle/
[48] http://worldsbiggestpacman.com/
[49] http://craftymind.com/factory/html5video/CanvasVideo.html
[50] http://codepen.io/binacio/pen/raiJI
[51] http://alteredqualia.com/canvasmol/
[52] http://www.kesiev.com/akihabara/
[53] http://9elements.com/io/projects/html5/canvas/

6. API's

Geolocalización

La geolocalización consiste en averiguar en qué lugar del mundo nos encontramos exactamente (mediante la dirección IP, conexión de red inalámbrica, torre de señal móvil, GPS, etc.).

HTML está desarrollando una API para que podamos detectar desde el navegador la posición exacta.

Por ejemplo[54]:

```
function get_location() {
    navigator.geolocation.getCurrentPosition(funcion);
}
```

Drag&Drop

Permite arrastrar y agarrar elementos en una página web. [55] [56]

Creación de elemento dragable:

```
<div class="elem" draggable="true">Elemento</div>
```

Detección de eventos de arrastre

Se deben tener en cuenta distintos eventos para controlar todo el proceso de arrastrar y soltar:

- dragstart
- drag
- dragenter
- dragleave
- dragover
- drop
- dragend

En combinación con Javascript[57], podremos darle diferentes utilidades[58] [59], como por ejemplo, un carrito de la compra[60], una ordenación de contenido[61], etc[62].

[54] http://html5demos.com/geo
[55] http://html5demos.com/drag
[56] http://code.makery.ch/library/dart-drag-and-drop/
[57] http://www.html5rocks.com/es/tutorials/dnd/basics/
[58] http://decafbad.com/2009/07/drag-and-drop/api-demos.html
[59] http://ljouanneau.com/lab/html5/demodragdrop.html
[60] http://nettutsplus.s3.amazonaws.com/64_html5dragdrop/demo/index.html
[61] http://decafbad.com/2009/07/drag-and-drop/outline.html
[62] http://web.ontuts.com/tutoriales/drag-drop-en-html-5/

Fullscreen (pantalla completa)

Permite controlar si la página se muestra a pantalla completa[63] [64](siempre con permiso previo del usuario) o no.

Primero se detecta el método correcto (cada navegador tiene el suyo, por estar en pruebas):

```
function launchIntoFullscreen(element) {
  if(element.requestFullscreen) {
    element.requestFullscreen();
  } else if(element.mozRequestFullScreen) {
    element.mozRequestFullScreen();
  } else if(element.webkitRequestFullscreen) {
    element.webkitRequestFullscreen();
  } else if(element.msRequestFullscreen) {
    element.msRequestFullscreen();
  }
}
```

Después, se lanza la pantalla completa (si el navegador lo soporta):

```
//Toda la página
launchIntoFullscreen(document.documentElement);

//Sólo un elemento concreto
launchIntoFullscreen(document.getElementById("videoElement"));
```

Y por último, se cierra:

```
function exitFullscreen() {
  if(document.exitFullscreen) {
    document.exitFullscreen();
  } else if(document.mozCancelFullScreen) {
    document.mozCancelFullScreen();
  } else if(document.webkitExitFullscreen) {
    document.webkitExitFullscreen();
  }
}

exitFullscreen();
```

Control de cámara, micrófono y video

Mediante las Api's correspondientes, podemos controlar la webcam[65] [66] de un usuario así como el micrófono[67], con el fin de, grabar video/audio y procesarlo, etc.

[63] https://fullscreen.spec.whatwg.org/
[64] https://davidwalsh.name/fullscreen
[65] http://davidwalsh.name/browser-camera
[66] http://html5-examples.craic.com/microphone_input_with_spectrogram.html
[67] https://developers.google.com/web/updates/2013/01/Voice-Driven-Web-Apps-Introduction-to-the-Web-Speech-API

Orientación

Podremos detectar la orientación y posición de un dispositivo[68].

Para ello, primero necesitaremos saber si hay soporte o no [69] [70]:

```
if (window.DeviceOrientationEvent) {
    console.log("DeviceOrientation is supported");
}
```

Batería

Podremos detectar el nivel de batería y ejecutar diferentes funciones al respecto.[71]

```
navigator.getBattery().then(function(result) {});
```

Resultado:

```
BatteryManagery {
    charging: false,
    chargingTime: Infinity,
    dischargingTime: 8940,
    level: 0.59,
    onchargingchange: null,
    onchargingtimechange: null,
    ondischargingtimechange: null,
    onlevelchange: null
}
```

Vibración

Controla la vibración del dispositivo[72] [73].

```
navigator.vibrate(1000);
```

Archivo y Sistema de archivos (File y FileSystem)

Permite leer/escribir archivos y directorios[74] [75].

[68] http://www.w3.org/TR/screen-orientation/
[69] http://www.html5rocks.com/en/tutorials/device/orientation/
[70] http://www.sitepoint.com/introducing-screen-orientation-api/
[71] https://davidwalsh.name/javascript-battery-api
[72] http://www.w3.org/TR/vibration/
[73] https://davidwalsh.name/vibration-api
[74] http://www.w3.org/TR/FileAPI/
[75] http://www.html5rocks.com/en/tutorials/file/dndfiles/

Validación Restringida (formularios)

Permite la validación personalizada de los campos de formulario [76] [77] [78].

willValidate

Indica si el elemento va a tener una validación restringida.

```
<input type="text" id="campo">

document.getElementById('campo').willValidate;
```

validity

Devuelve el estado de la validación (varias propiedades)

validationMessage

Contiene el mensaje que el navegador mostrará al usuario

checkValidity()

Devuelve “true” si el elemento contiene datos válidos.

```
document.getElementById('campo').checkValidity();
```

setCustomValidity()

Permite modificar el mensaje de validación y las reglas de validación en sí mismas.

```
if (document.getElementById('password1').value !=
document.getElementById('password2').value) {

document.getElementById('password1').setCustomValidity('Passwords
must match.');
} else {
    document.getElementById('password1').setCustomValidity('');
}
```

[76] https://html.spec.whatwg.org/#constraint-validation-api
[77] http://www.html5rocks.com/en/tutorials/forms/constraintvalidation/
[78] http://www.sitepoint.com/html5-forms-javascript-constraint-validation-api/

Web Messaging

Permite la comunicación segura entre documentos de diferente origen/ dominio[79] [80] [81].

WebRTC

Comunicación en tiempo real entre Navegadores[82] [83] [84]

Web Workers

Permite la ejecución de un script javascript en segundo plano [85] [86] [87].

Web Sockets

Proporciona un canal de comunicación bidireccional y full-duplex sobre un único socket TCP.[88] [89] [90]

Almacenamiento (local, sesión y base de datos)

Mediante los nuevos sistemas de almacenamiento de HTML 5, vamos a poder almacenar información en el navegador.

Las cookies, que actualmente utilizamos, están pensadas para almacenar una información escasa y los navegadores las envían al servidor cada vez que se recarga la página, por lo que se consume tiempo y ancho de banda extras.

Por lo tanto, se necesita un nuevo sistema para poder almacenar gran cantidad de información y que el intercambio de esta información entre el navegador y el servidor sea rápida y eficaz.

Existen 3 tipos de almacenamiento[91] [92]:

[79] https://html.spec.whatwg.org/multipage/comms.html#web-messaging
[80] https://html.spec.whatwg.org/multipage/comms.html
[81] https://dev.opera.com/articles/window-postmessage-messagechannel/
[82] http://w3c.github.io/webrtc-pc/
[83] https://developer.mozilla.org/en-US/docs/Web/API/WebRTC_API/Taking_still_photos
[84] http://www.html5rocks.com/en/tutorials/webrtc/basics/
[85] http://www.w3.org/TR/workers/
[86] https://developer.mozilla.org/es/docs/Web/Guide/Performance/Usando_web_workers
[87] http://html5demos.com/worker
[88] http://www.html5rocks.com/es/tutorials/websockets/basics/
[89] https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API
[90] https://www.websocket.org/demos.html
[91] https://html.spec.whatwg.org/multipage/webstorage.html
[92] https://developer.mozilla.org/en-US/docs/Web/API/Storage

Local [93]*:*

para almacenar datos (sólo pares clave/valor) en la máquina del usuario. Los datos almacenados son únicos al dominio (preferencias).

```
localStorage.setItem('nombre_clave', 'valor');

myStorage = localStorage;
```

Sesión [94]*:*

Para almacenar datos (sólo pares clave/valor) únicamente válidos durante la sesión (carritos de la compra o estado de aplicación).

```
sessionStorage.setItem('nombre_clave', 'valor');

var data = sessionStorage.getItem('nombre_clave');
```

Base de datos (IndexedDB) [95]*:*

para almacenar datos relacionales ofreciendo una API de base de datos SQL, con todo lo que ello implica[96].

Con el tipo de almacenamiento adecuado, podremos desarrollar nuestras aplicaciones fácilmente.

Algunos casos de uso son presentaciones[97], juegos, etc..

[93] https://developer.mozilla.org/en-US/docs/Web/API/Window/localStorage
[94] https://developer.mozilla.org/en-US/docs/Web/API/Window/sessionStorage
[95] http://www.w3.org/TR/IndexedDB/
[96] https://developer.mozilla.org/en-US/docs/Web/API/IndexedDB_API/Using_IndexedDB
[97] http://lab.hakim.se/reveal-js

PARTE 2: SOPORTE

1. SOPORTE EN NAVEGADORES

Los navegadores que aún no soportan HTML 5 van a necesitar una ayuda para que rendericen correctamente los elementos.

Detección de soporte HTML 5

* *Modernizr*[98]: librería JavaScript con licencia MIT que detecta la compatibilidad de nuestro navegador con HTML5 y CSS3

Detección de soporte para `autofocus`[99], por ejemplo.

También podemos conocer si nuestro navegador actual[100] soporta HTML 5

Declaración CSS:

```
/* Declarando elementos HTML 5 */
article,aside,canvas,details,figcaption,figure,footer,header,hgro
up,menu,nav,section,summary{  display: block;  }
```

Para IE:

* HTML5 Shiv [101] [102]

```
<!--[if lt IE 9]><script
src="//html5shiv.googlecode.com/svn/trunk/html5.js">
</script>
<![endif]-->
```

* IE Canvas[103]

[98] http://www.modernizr.com/
[99] http://diveintohtml5.info/detect.html#input-autofocus
[100] http://html5test.com/
[101] http://code.google.com/p/html5shiv/
[102] http://remysharp.com/2009/01/07/html5-enabling-script/
[103] http://code.google.com/p/explorercanvas/

2. MÁS RECURSOS

Polyfills:

https://github.com/Modernizr/Modernizr/wiki/HTML5-Cross-browser-Polyfills

Tablas de soporte:

http://caniuse.com/

http://html5please.com

http://html5test.com/

Enlaces indispensables:

https://css-tricks.com

https://www.smashingmagazine.com

Otros enlaces:

https://developer.mozilla.org/en/HTML/HTML5

http://dev.opera.com/articles/tags/html5/

http://html5doctor.com

http://wufoo.com/html5/

http://diveintohtml5.info

http://www.w3conversions.com/resources.html (prácticamente están todos los recursos unificados)

Recopilación de demos:

http://html5demos.com/

https://www.chromeexperiments.com/