

HTML 5

Historia del HTML 5 y sus antecesores.

Desde que IBM empezó a utilizar su lenguaje GML hasta el momento actual en el que nos encontramos han sido muchos cambios tecnológicos que han ido produciéndose. Con la llegada del lenguaje GML transformándose en estándar formal en 1986 (Standard Generalized Markup Language) IBM proporcionó un lenguaje que definía como debían ser los lenguajes de marcado pero no especificaba como tenían que ser las etiquetas. Es cuando a partir del SGML empieza a desarrollarse el HTML con los siguientes hitos importantes en su línea de tiempo:

- 1995: Se formaliza el HTML 2.0 y con ello la sintaxis y la mayoría de las reglas que se encuentran actualmente implementadas.
- 1997: HTML 3.2 durante mucho tiempo ignorado por las empresas que proveen navegadores, las cuales implementan sus propias etiquetas.
- 1998: Presionadas por la adopción de los estándar web, se otorga peso a las recomendaciones del W3C y se promocionan navegadores basados en dichos estándar.
- 1999: Se estabiliza la sintaxis y la estructura del HTML 4.0, convirtiéndose en el estándar para la web.
- 2000: Nace el XHTML 1.0 diseñado para adaptar el HTML a XML. Uso de DTD para renderizar como HTML.
- 2000-2004: El incremento de las conexiones en el ancho de banda es alto, produciéndose una demanda en el campo del desarrollo de las aplicaciones y la multimedia, donde tecnologías como Flash y Ajax hacen que se trabaje en la especificación XHTML 2.0.
- 2004: No satisfechos con la dirección de XHTML, Apple, Mozilla y Opera proponen evolucionar el estándar HTML 4.0. Aunque son rechazados, forman el WHATWG (Web Hypertext Application Technology Working Group).
- 2005: Se publica el borrador de trabajo Web Applications 1.0.
- 2007: El W3C adopta el trabajo de WHATWG en un nuevo capítulo, publicando lo que sería el borrador de trabajo de HTML 5.
- 2009: Última llamada expedido para el proyecto de trabajo de HTML 5. El W3C no renueva XHTML 2.0.
- 2010: Borrador del W3C para HTML5.

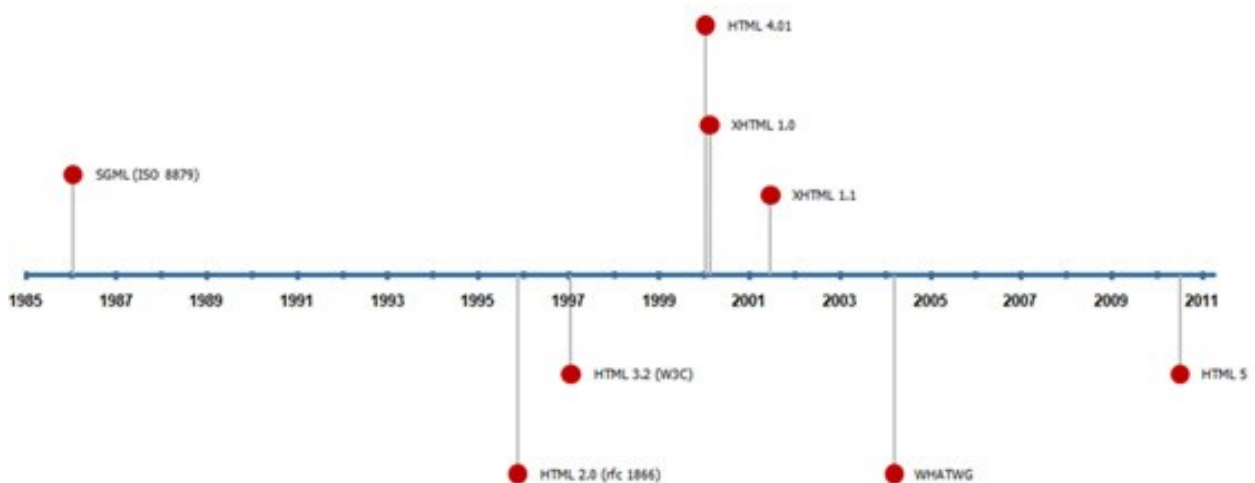


Imagen 1: Evolución del lenguaje de marcado HTML

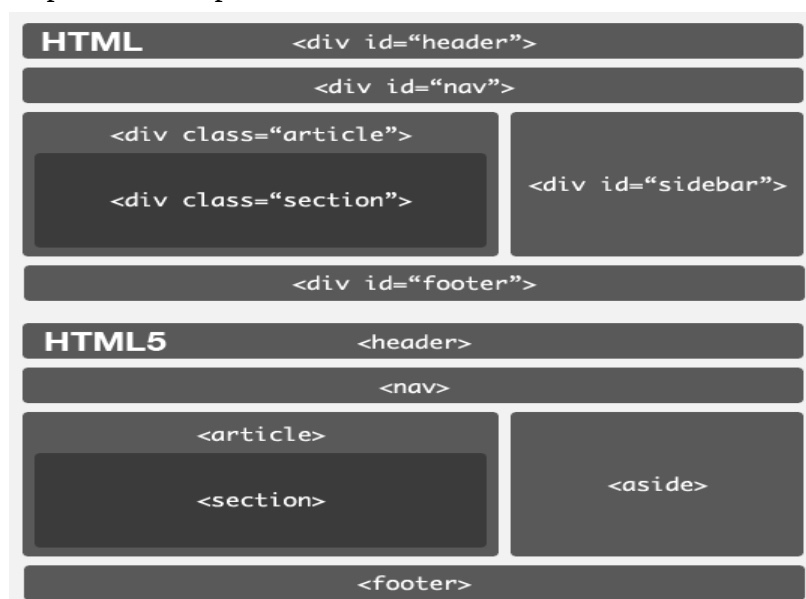
Características y Novedades Principales.

El enfoque general ha cambiado bastante respecto a versiones anteriores de HTML, añadiendo semántica y accesibilidad implícitas, especificando cada detalle y borrando cualquier ambigüedad. También se tiene en cuenta que muchas páginas web actuales son dinámicas, pareciéndose más a aplicaciones que a documentos. Algo básico es que HTML5 está definido en base al DOM (la representación interna de una web con la que trabaja un navegador), dejando de lado la representación “real”, definiendo a la vez un estándar HTML y XHTML.

- **Mejor estructura:**

Hoy en día se abusa bastante del elemento `div`, que nos permite estructurar una web en bloques. En HTML5 hay varios elementos que sirven para estructurar mejor una página web, estableciendo qué es cada sección, y reemplazando en muchas ocasiones a `div`. Con este extra de semántica, será mucho más coherente y fácil de entender por otras personas. Y lo que es más importante, será trivial de entender para una máquina, dándole más importancia a unas secciones y pudiendo jugar con esos datos automáticamente. Concretamente, la tarea de un buscador será mucho más fácil, pero cualquier aplicación que “lea” páginas web se beneficiará. Estos son los elementos:

- **section** representa una sección “general” dentro de un documento o aplicación, como un capítulo de un libro. Puede contener subsecciones y si lo acompañamos de `h1-h6` podemos estructurar mejor toda la página.
- **article** representa un contenido independiente en un documento, el caso más claro son las entradas de un blog o las noticias de un periódico online. Así, dentro de la portada podremos tener varios artículos demarcados semánticamente, por lo que una herramienta puede extraerlos fácilmente.
- **aside** representa un contenido que está muy poco relacionado con el resto de la página, como una barra lateral. Esencial para delimitar el contenido “importante” del contenido “de apoyo”, haciendo más caso al primero que al segundo.
- **header** representa la cabecera de una sección, y es de suponer que se le dé más importancia que al resto, sobre todo si la sección es un artículo.
- **footer** representa el pié de una sección, con información acerca de la página/sección que poco tiene que ver con el contenido de la página, como el autor, copyright, etc.
- **nav** representa una sección dedicada a la navegación entre el sitio, como la típica barra superior de los periódicos.



- **Mejores formularios:**

El elemento input ha sido ampliado y ahora permite todos estos tipos de datos:

- **datetime, datetime-local, date, month, week, time**, para que indicar una fecha/hora.
- **number** para que el usuario indique un número.
- **range** para indicar un rango entre dos números.
- **email** para indicar un correo electrónico.
- **url** para indicar una dirección web.
- **search** para indicar una búsqueda.
- **color** para indicar un color.

Lo más interesante de esto es que los navegadores podrán implementar interfaces específicas para cada tipo de dato, por ejemplo una fecha o un color se podrán indicar de manera directa e intuitiva. Otro ejemplo sería el teclado del iPhone, que muestra unos símbolos u otros dependiendo de si es un texto normal, un email (añade @ y el punto) o una url (añade la barra y el punto com), y que por tanto gana mucho con este estándar.

- **Otros elementos importantes:**

- **audio y video** sirven para incrustar un contenido multimedia de sonido o de vídeo, respectivamente. Sin duda uno de los añadidos más interesantes, ya que permite reproducir/controlar vídeos y audios sin necesidad de plugins como el de Flash. Se tratan de manera totalmente nativa como cualquier otro elemento, por ejemplo se pueden incluir enlaces o imágenes dentro de un vídeo. Aunque las implementaciones actuales son un tanto ineficientes, se espera que en un futuro próximo se optimicen. Portales de vídeo como Youtube o Dailymotion ya están empezando a mostrar que un futuro sin Flash es posible (¡y necesario!).
- **embed** sirve para contenido incrustado pero no nativo, sino ejecutado por plugins como el de Flash. Aunque embed está soportado por casi todos los navegadores desde hace tiempo, es ahora cuando entra parte del estándar y evita el infierno/pelea entre object y embed.
- **canvas** es un elemento complejo que permite generar gráficos, dibujando elementos dentro de él. Aunque nunca hayas oído hablar de él, seguro que lo has usado alguna vez, por ejemplo de Google Maps. Es un elemento muy potente que dará bastante que hablar en el futuro, y que será el culpable de aplicaciones web espectaculares.

- **Más elementos:**

- **dialog** se plantea para escribir conversaciones, por ejemplo para transcripciones de chat.
- **figure** se plantea para asociar un contenido multimedia (una foto, un vídeo, etc) a un título o leyenda.
- **mark** representa un texto resaltado, por ejemplo para resaltar una búsqueda.
- **meter** representa una medida, como el número de KB. Tiene más sentido si lo unimos con...
- **progress** representa el estado de una tarea, y se puede usar por ejemplo al subir un documento o al realizar varias tareas pesadas. Esto permitirá barras de tareas personalizadas y potentes.
- **time** representa una fecha o una hora.
- **output** representa una salida de un programa, probablemente ejecutado directamente en el navegador, como una calculadora.
- **datagrid** representa datos de manera interactiva y permite trabajar dinámicamente con información y cambiar la página respecto a esa información. Será útil sobre todo si se quiere trabajar con aplicaciones que necesiten de bastantes datos a la vez en el lado del cliente.

- **Nuevas APIs:**

No está claro que todas las APIs siguientes se vayan a incluir en el estándar HTML5 propiamente, de hecho seguro que alguna de ellas se separará creando un estándar propio dedicado. De cualquier forma, estas son las nuevas APIs que nacen o se desarrollan en HTML5:

- Una API para dibujar en 2D, que se podrá usar junto al nuevo elemento canvas. Básicamente, se pueden pintar elementos sobre un lienzo, de manera similar a lenguajes como Java.
- Una API para controlar los nuevos elementos multimedia, video y audio. Así podremos controlar la reproducción con código Javascript, algo interesante pero que puede dar más de sí.
- Una API para guardar datos localmente, utilísimo para que las aplicaciones web puedan trabajar sin necesitar conexión a Internet. Ese DOM storage está ya implementado en las últimas versiones de los grandes navegadores, así que dentro de poco podremos disfrutar de esta clase de aplicaciones sin necesidad de extensiones como Google Gears.
- Una API para que las aplicaciones web puedan enlazarse a protocolos o tipos de archivos MIME, otro añadido extremadamente útil. Esto permitiría abrir las fotos de tu disco duro directamente en una aplicación de retoque online, o un archivo mp3 en una biblioteca online, etc...
- Una API para editar los campos que sean editables, valga la rebuznancia. Esto permite controlar los elementos HTML que son editables por el usuario, tipo Word. Por ejemplo, Google Docs o editar HTML en emails será muy sencillo, y con esta API es trivial cambiar ese contenido con Javascript.
- Una API para controlar las acciones Drag & Drop sobre los elementos que se puedan arrastrar. Esto se puede conseguir actualmente con algunas librerías, pero con esta API el navegador lo permite de manera nativa y más poderosa.
- Una API para controlar el historial desde una aplicación web. Esto permitirá a las aplicaciones web que se muevan con Javascript añadir páginas al historial para que los botones Atrás-Adelante funcionen siempre.
- Una API para habilitar la comunicación entre varias “páginas web”. Es decir, si tenemos varios iframes externos en una web, podemos comunicarnos con ellos y compartir información de manera segura, por ejemplo con gadgets de Facebook o similares.

- **Novedades en el DOM:**

Se han añadido a los elementos del DOM nuevas funciones y atributos que facilitan su uso y permiten realizar acciones muy usadas. Aquí comentaré los más interesantes, que trabajan sobre el documento (HTMLDocument) o sobre cualquier elemento (HTMLElement).

- La función `getElementsByClassName()` se añade a todos los elementos y al documento. Su funcionamiento es similar a `getElementsById()`, pero en este caso selecciona todos los elementos del documento o de cierto subárbol del documento que contengan esa clase. Su definición es tan amplia que incluso elementos que contenga SVG o MathML pueden ser encontrados.
- El atributo `classList` está disponible para cualquier elemento, y contiene una lista con todas las clases que tiene ese elemento. Además contiene varios métodos que facilitan trabajar con esa lista: buscar, modificar, borrar, etc. Muy útil para trabajar con elementos que puedan tener más de una clase, es muy sencillo y muy conveniente de usar. Los enlaces también tienen un elemento similar adicional llamado `relList` que permite lo mismo pero con el atributo `rel` (como el famoso `rel="nofollow"`).
- El atributo `innerHTML` se añade, por fin, al estándar. Prácticamente usado por todas las aplicaciones web y soportado por todos los navegadores, creo que todos los desarrolladores web lo conocen de sobra. Para aquellos que no lo conozcan, este atributo

permite trabajar con el contenido de un elemento en texto plano, incluso cambiando elementos HTML que pueda haber. Igualmente, se añade a todos los elementos y al propio documento, pudiendo cambiar TODO el contenido de una web.

- Los atributos `activeElement` y `hasFocus` están disponibles sobre un documento, y permiten conocer qué elemento está activo y si el documento tiene el foco, respectivamente.
- La función `getSelection()` se aplica también al documento y devuelve un objeto con el texto y elementos que están seleccionados.
- El atributo `designMode` es otra novedad sobre el documento e indica/modifica que el documento pueda o no ser editado.
- La función `execCommand()` se aplica sobre el documento y permite ejecutar acciones o “comandos” típicos de edición de documentos. Por ejemplo, con este método se llamaría a los útiles copiar/pegar, pero también a otros típicos como crear un enlace o cambiar el color de un elemento. Como el anterior, la mayoría de estos comandos trabajan sobre elementos editables.

- **Elementos eliminados:**

Como decía, estos son los elementos eliminados y las razones de por qué son prohibidos.

- Los siguientes elementos (muy usados hace pocos años) se quitan de HTML5 porque son puramente presentacionales (no tienen semántica) y todo el tema estético se debe tratar con CSS:
 - `basefont`
 - `big`
 - `center`
 - `font`
 - `s`
 - `strike`
 - `tt`
 - `u`
- **Los elementos para trabajar con frames** (`frame`, `frameset` y `noframes`) se quitan del estándar por razones obvias: afectan negativamente a la usabilidad y accesibilidad de la web. Además, prácticamente rompen la web, y si se necesita algo similar se puede acudir a los `iframe`, más potentes y mejor pensados.
- **El elemento `acronym`** se elimina simplemente porque crea confusión sobre su uso, y los desarrolladores no entienden demasiado bien para qué usarlo. Las abreviaciones y acrónimos se pueden marcar con `abbr`, que sí se mantiene en el estándar.
- **El elemento `applet`** se ha declarado obsoleto y hoy en día no se utiliza. El elemento `object` reemplaza sus funciones y es lo común hoy en día.
- **El elemento `isindex`** se quita definitivamente. En la era de las cavernas se utilizaba para mandar información al servidor, pero con la llegada de los formularios su uso es arcaico y poco útil.
- **El elemento `dir`** también se declara obsoleto (ya lo era en HTML4), y simplemente se recomienda usar listas normales con `ul`.
- **El elemento `noscript`** se mantiene en HTML pero no en XML/XHTML, ya que su contenido está en HTML. No estoy muy de acuerdo con este movimiento, pero así será.

Diferencias entre HTML5 y sus predecesores.

En el siguiente [enlace](#) y en el apartado anterior se encuentran las diferencias entre el HTML5 y sus predecesores.

- **Sintaxis**

El término HTML posee una sintaxis compatible con HTML4 y XHTML1 publicados en la red, pero no compatible con las características más esotéricas del SGML de HTML4.

- **Codificación de caracteres**

Al igual que sus hermanos menores, seguiremos pudiendo definir el charset de nuestro documento mediante el tag `<meta charset="UTF-8" >` o la correspondiente versión de XML para XHTML5.

- **El DOCTYPE**

El nuevo HTML5 requiere el elemento DOCTYPE que debe ser declarado al principio de la página, de esta forma nos aseguramos de que el navegador renderiza la página en modo estándar. En cambio para la versión XHTML5 este elemento es opcional debido a que XML actúa de diferente manera dentro de nuestro navegador.

- **Modelos estrictos de contenidos**

HTML5 define de forma más estricta el contenido para elementos `<div />` y ``. Estos elementos ahora pueden contener contenido de elementos “block” o “inline” pero no los dos. En HTML4 esto fue considerado como un bug de la especificación ya que permitía el uso de ambos.

- **Nuevos elementos**

Los tiempos modernos requieren nuevos elementos para proporcionar una web más semántica, completa y homogénea. Para ello se han añadido una buena serie de elementos que nos permitirán encapsular más nuestro contenido.

- **Nuevos atributos**

HTML 5 ha introducido una gran cantidad de nuevos elementos para varios elementos de los que ya disponemos en la HTML4.

- **Extensión de HTMLDocument**

HTML5 también ha modificado el elemento padre del DOM Level 2. En él encontramos una serie de mejoras y otras que finalmente se hacen estándares:

- **getElementsByClassName()**, para seleccionar elementos por el atributo class. Ya lo comentamos hace tiempo y vimos que las diferencias a nivel de tiempo de respuesta eran más que satisfactorias.
 - **innerHTML**, aunque prácticamente se usa en todas, o casi todas, las aplicaciones web existentes, por fin será reconocido como estándar en la especificación. Además aprovechando si inserción se posibilita su uso en el elemento padre.
 - **activeElement**, **hasFocus()**, nos permitirá conocer el elemento activo en tiempo real y el que tenga el foco.
 - **getSelection()**, devuelve un objeto con la selección actual.
 - **designMode** y **execCommand()**, muy usados para editar documentos.

Ventajas e inconvenientes de HTML5.

Las ventajas están claras y ya se han expuesto en los puntos anteriores, como clara desventaja quedaría remarcar la lenta adaptación al nuevo estándar y que todavía quedan unos años hasta que HTML5 está extendido.

Contenido multimedia.

- Video y Audio. (ventajas e inconvenientes frente a flash).
- Controles multimedia (src, preload, autoplay, loop, controls, etc).

El tag <video /> es sin duda el atractivo más visual de este nuevo estandar y es que su llegada puede revolucionar el mundo multimedia en Internet.

HTML 4

```
<object classid="clsid:d27cdb6e-ae6d-11cf-96b8-444553540000" width="425" height="344"
codebase="http://download.macromedia.com/pub/shockwave/cabs/flash/swflash.cab#version=6,0,4
0,0">
<param name="allowFullScreen" value="true" />
<param name="allowscriptaccess" value="always" />
<param name="src" value="http://www.youtube.com/v/oHg5SJYRHA0&hl=en&fs=1&" />
<param name="allowfullscreen" value="true" />
<embed type="application/x-shockwave-flash" width="425" height="344"
src="http://www.youtube.com/v/oHg5SJYRHA0&hl=en&fs=1&" allowscriptaccess="always"
allowfullscreen="true">
</embed>
</object>
```

La forma de visualizar un video en una página web, actualmente es similar a la que se muestra en este código. Pasa por el elemento <object /> que nos permite especificar otro <embed /> que nos permite incrustar una película flash en nuestro página.

Semánticamente, estamos hablando de un objeto (<object />) embebido en la página (<embed />), pero no nos dice que se trata de un video ya que este código, sirve para incrustar juegos flash, galerías,....

HTML5

```
<video width="640" height="360" src="http://www.youtube.com/demo/google_main.mp4"
controls autobuffer>
<p> Try this page in Safari 4! Or you can <a
href="http://www.youtube.com/demo/google_main.mp4">download the video</a> instead.</p>
</video>
```

Con HTML5, además de quedar más claro que este contenido es un contenido de formato video nos permite informar una alternativa a los usuarios que no estén visualizando la aplicación con un navegador con esta capacidad. Posibilitando descargar el video para que el usuario lo descargue y lo visualice con su reproductor predeterminado.

Actualmente hay una guerra abierta sobre que codec utilizar para la reproducción de video, por un lado está el codec H.264 (propietario) con una supuesta calidad superior y por otra Ogg Theora (libre) con calidad supuestamente inferior, y los navegadores se han alineado al lado de uno u otro.

Esto nos da como resultado que la calidad nativa de los codecs de video es muy superior a la de Flash, y el uso del procesador es muy inferior, dejando en clara desventaja a flash, que no fue creado para estos menesteres, pero que hasta la actualidad ha suplido esta carencia.

Del mismo modo en el caso del audio:

El tag <audio />

Destinado para la inclusión directa mediante HTML de ficheros de audio en nuestras páginas web este tag, evitará que tengamos que usar reproductores creados el flash para que nuestros usuarios puedan escuchar sonidos, canciones, podcasts... directamente desde nuestra página web.

```
<audio src="music.oga" controls>
  <a href="music.oga">Descargar canción</a>
</audio>
```

Como podemos ver, además de delegar al navegador la reproducción del fichero, hacemos nuestro código más semántico aplicando una mayor claridad al elemento usado, mucho más que como lo estábamos haciendo hasta ahora:

```
<object type="application/x-shockwave-flash" data="player_mp3.swf" width="200" height="20">
  <param name="movie" value="player_mp3.swf" />
  <param name="FlashVars" value="mp3=music.mp3&showstop=1&showinfo=1" />
</object>
```

Al ser un tag abierto (<audio></audio>) nos permite ofrecer un valor alternativo para los usuarios que no tienen disponible esta capacidad. En el primer ejemplo vemos como ofrecemos la posibilidad de descargar el fichero de audio music.oga en caso de que el navegador no reconozca el tag <audio />.

```
<audio src="music.oga" controls>
  <object type="application/x-shockwave-flash" data="player_mp3.swf" width="200" height="20">
    <param name="movie" value="player_mp3.swf" />
    <param name="FlashVars" value="mp3=music.mp3&showstop=1&showinfo=1" />
  </object>
</audio>
```

Aprovechándonos de esta propiedad, podemos hacer una mezcla entre las dos técnicas para así acercar ambas técnicas a todos los usuarios. Eso sí, a costa de no validar nuestra página.

Controles multimedia:

se trata de parametros que permiten configurar el objeto.

Video:

Atributo	Valor	Descripcion
audio	muted	Indica el estado del audio, actualmente solo "muted" es soportado.
autoplay	autoplay	El video se reproducira tan pronto como este preparado/cargado.
controls	controls	permite mostrar controles sobre el video.
height	pixels	permite indicar el "height" del video.
width	pixels	permite indicar el "width" del video.
loop	loop	permite el "looping" del video de forma indefinida.
poster	url	indica una URL de la imagen del video.
src	url	La URL del video.
preload	preload	El video se cargara a la vez que la pagina y se ejecutara tal y como se cargue, este atributo se ignora si esta el atributo autoplay.

Audio:

Atributo	Valor	Descripcion
autoplay	autoplay	El audio se reproducira tan pronto como este preparado.
controls	controls	permite mostrar controles sobre el audio.
loop	loop	permite el "looping" del audio de forma indefinida.
src	url	La URL del audio.
Preload	auto, metadata, none	

Bibliografia:

<http://www.anexom.es/tecnologia/disenio-web/las-novedades-de-html5-i/>
<http://www.desarrolloweb.com/articulos/que-es-html5.html>
http://es.wikipedia.org/wiki/HTML_5#Novedades
<http://www.desarrolloweb.com/actualidad/html-4-5-diferencias-1465.html>
<http://www.w3.org/TR/2009/WD-html5-diff-20090212/>
<http://www.anieto2k.com/2007/06/16/las-principales-diferencias-entre-html5-y-html4/>
<http://www.anieto2k.com/2009/06/30/el-elemento-video-del-html5/>
<http://www.rubendomfer.com/blog/2010/03/01/video-en-html-5/>
http://en.wikipedia.org/wiki/HTML5_video
http://www.w3schools.com/html5/tag_audio.asp
http://www.w3schools.com/html5/html5_audio.asp
<http://www.anieto2k.com/2009/07/14/el-elemento-audio-del-html5/>