1. REPRODUCIR SIN PLUGIN

Los navegadores siempre han sido capaces de mostrar contenido estático, como texto, formularios o imágenes, pero han tenido más dificultades con contenido interactivo y multimedia. Para esto último han confiado en los complementos o *plugins* externos.

Flash ha sido la tecnología más utilizada para incorporar vídeo y audio en una página web, principalmente porque casi todo el mundo tiene instalado el componente **Flash Player** en su ordenador (al menos en el mundo Windows).

Y ahora aparece **HTML5** incorporando nuevos elementos para reproducir **vídeo** y **audio** como parte integrante del lenguaje.

Esto quiere decir que un navegador compatible con esas características de HTML5 ya no necesita de un complemento para reproducir vídeo o audio. Es el propio navegador el que incorpora esa funcionalidad.

De esta forma, el usuario ya no tiene que preocuparse si dispone de la versión necesaria del complemento para reproducir el contenido multimedia o si tiene que instalarlo.

Sin lugar a dudas, esto es un gran avance, pero todavía no es una tecnología lo suficientemente madura para pensar que ya no necesitamos Flash.

El problema principal lo encontramos a la hora de elegir el **formato** y **códec** con el que proporcionar los archivos de vídeo y audio. Al principio, la propia especificación HTML5 incluía formatos específicos, pero más tarde se eliminaron esas referencias debido a reticencias por parte de algunos fabricantes.

Actualmente conviven distintos formatos y códecs y, lo que es peor, los navegadores no siempre incorporan los mismos.

Esto es un problema para el diseñador web, ya que se ve en la obligación de disponer de más de una versión del mismo archivo de vídeo o de audio.

Además, si desea que también funcione en navegadores antiguos, que no son compatibles con esas funcionalidades de HTML5, debe incorporar una versión adicional en formato Flash.

A medida que avance el uso de HTML5 todo esto irá clarificándose.

El vídeo y audio Flash seguirá siendo utilizado para proporcionar compatibilidad con los navegadores antiguos y porque es una tecnología más avanzada, que actualmente proporciona más funcionalidad y que dispone de un gran número de diseñadores que la utilizan. Cuando HTML5 alcance esa madurez, ¿quién sabe?

2. LA ETIQUETA <VIDEO>

Para incluir vídeo con HTML5 en una página web utilizaremos la etiqueta **<video>**. Esta etiqueta es bastante sencilla, aunque puede complicarse al incluir archivos alternativos de vídeo para conseguir la compatibilidad con el mayor número de navegadores y dispositivos móviles.

En su versión sencilla la etiqueta incluye únicamente el atributo **src**. Este atributo, de forma similar a lo que ocurre con la etiqueta **<img>**, sirve para indicar la ubicación del archivo de vídeo que se reproducirá en la página web:

```
<video src="media/coches.mp4"></video>
```

Con esto es suficiente para reproducir un vídeo. Sin embargo, normalmente incluirás código "alternativo" para aquellos navegadores que no son compatibles con el elemento **<video>** de HTML5.

Por ejemplo, una forma sencilla de hacerlo es incluir un enlace para que el usuario pueda descargarse el vídeo (seguramente para reproducirlo con una aplicación externa) o simplemente utilizar una imagen significativa del vídeo en lugar de este:

```
<video src="media/coches.mp4">
Descárguese el vídeo desde
<a href="media/coches.mp4">aquí</a>.
</video>
```

También es habitual incluir otros atributos en la etiqueta **video**, aunque el único necesario es **src**. Por ejemplo:

- **width** y **height**: para establecer las dimensiones del vídeo. Si no coinciden con las dimensiones originales, el navegador puede ajustar el tamaño estirando o reduciendo el vídeo.

- **autoplay**: indica que el vídeo debe ser reproducido automáticamente, sin necesidad de que el usuario pulse en el correspondiente botón de reproducción.

Además, utilizando autoplay se provoca que se descargue el vídeo, empleando el ancho de banda de la conexión (si no se trata de una tarifa plana, esto puede significar un coste para el usuario que realmente no desea ver el vídeo).

Este atributo debe utilizarse con mucho cuidado, ya que es molesto para el usuario que el vídeo empiece si realmente no quiere verlo.

Por otra parte, tiene sentido si el contenido fundamental de la página web es ese vídeo. En ese caso, se supone que si el usuario ha solicitado la página es para visualizar el vídeo.

Así pues, incluiremos este atributo únicamente cuando queramos utilizarlo. Simplemente tienes que escribir **autoplay** en la etiqueta de inicio.

- **controls**: permite que se muestre controles para la reproducción y pausa del vídeo.

Desde luego, si no se utiliza **autoplay**, la única forma que tiene el usuario para reproducir el vídeo es si se muestran estos controles.

El aspecto de estos controles puede diferir entre un navegador u otro.

- **poster**: permite incluir una imagen estática como marcador del vídeo, es decir, que se utiliza hasta que se reproduce el vídeo.

Si no se incluye, entonces el navegador utilizará el primer fotograma del vídeo. Esto tiene el problema de que puede ser un fotograma poco significativo, por lo que se da la oportunidad de cambiarlo mediante el atributo **poster**.

- **loop**: determina si el vídeo se repetirá indefinidamente.
- **preload**: cuando se utiliza, el navegador empieza a descargar el vídeo inmediatamente pero no lo reproduce.

El atributo **preload** tiene tres posibles valores:

- **preload=auto** (o simplemente **preload**): sugiere que el navegador debe empezar a descargar el vídeo inmediatamente.
- **preload=none**: **sugiere** que el navegador no debería descargar el vídeo hasta que el usuario lo solicitase.
- **preload=metadata**: **sugiere** que el navegador debería empezar a descargar únicamente información acerca del vídeo (duración, dimensiones, etc.). Las distintas pistas de audio y vídeo que visualiza el usuario solo deberían descargarse cuando este "le dé al play".

Este atributo puede utilizarse cuando estamos bastantes seguros de que el usuario visualizará el vídeo. De esta forma, nos adelantamos a su acción descargando el vídeo con antelación.

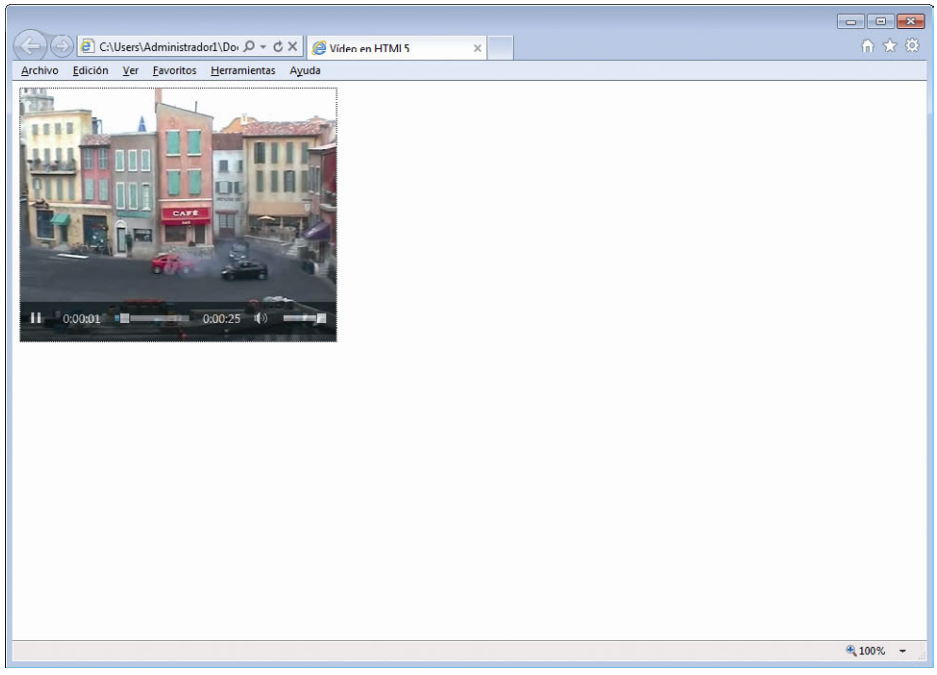
Por ejemplo, podríamos tener la siguiente etiqueta <video>:

```
<video src="media/coches.mp4" width="350" height="280"
controls poster="media/poster.jpg">
Descárguese el vídeo desde
<a href="media/coches.mp4">aquí</a>.
</video>
```

Si reproduces este vídeo en Internet Explorer, no tendrás problema, como puedes ver en la figura siguiente; sin embargo, en Firefox no se reproducirá.

Esto se debe a que sí que es compatible con la etiqueta **<video>** pero no con el formato y códec que se ha utilizado.

Así pues, hay que tener en cuenta los formatos y códecs de vídeo.



Si al reproducir el vídeo en un navegador no puedes ver los controles, sitúa el puntero del ratón por encima del vídeo. Para el caso de MP4, la imagen de poster solo se muestra un momento.

3. FORMATOS Y CÓDECS DE VÍDEO

Como se ha comentado, en las primeras versiones de la especificación del HTML5 se hacía referencia a un determinado códec de vídeo y de audio. Exactamente al códec **Ogg Theora** para el vídeo y al códec **Ogg Vorbis** para el audio.

Ambos códecs son libres por lo que los navegadores podrían implementarlos sin necesidad de pagar licencias adicionales.

Sin embargo, algunos fabricantes no estuvieron de acuerdo con utilizar únicamente este par de códecs, presentando sus propias alternativas.

Esto provocó que se eliminaran de la especificación de HTML5 las referencias a los códecs **Ogg** y no se indicara ninguno en concreto.

Puedes encontrar más información acerca de esta historia (en inglés) aquí:

http://en.wikipedia.org/wiki/HTML5_video#Supported_video_formats

El problema se plantea cuando los navegadores implementan sus propios conjuntos de formatos y códecs ya que, por supuesto, no lo hacen igual.

¿Qué es un **formato** de vídeo o de audio? ¿Qué es un **códec**?

El **formato** de un archivo de vídeo es el contenedor de la información, por lo que establece cómo se almacena el contenido del vídeo.

Así, tenemos **formatos** populares de vídeo para la Web como **MPEG4**, **Matroska**, **Ogg**, **webM** y **Flash** (FLV).

Por su parte, los **códecs** se encargan de comprimir y descomprimir la información del vídeo para reducir su tamaño. El propósito de un códec siempre es conseguir el menor espacio en disco pero con la mejor calidad del vídeo posible.

Por lo tanto, el navegador únicamente tiene que implementar la parte para "*descomprimir*" de un códec.

Algunos códecs populares de vídeo son:

- Para el contenedor **MPEG4** se utiliza el códec **H.264**.
- Para el contenedor **Ogg** se utiliza el códec **Ogg Theora**.
- Para el contenedor **Flash** se pueden utilizar los códecs **H.264** y **VP8**.
- Para el contenedor **webM** se utiliza el códec **VP8**.

En todos los casos, estos formatos y códecs admiten el *streaming* o descarga progresiva del vídeo, lo que los hacen adecuados para las páginas web.

Además, un mismo archivo de vídeo puede incluir también pistas de audio. Esto significa que en el mismo formato podríamos aplicar más de un códec para cada pista.

Lo importante aquí es la compatibilidad entre los distintos formatos y códecs que proporcionan los navegadores. Actualmente esto provoca bastante confusión, ya que:

- **Internet Explorer** soporta el códec **H.264** y, si está instalado en el sistema operativo, también el códec **VP8**.
- **Firefox** y **Opera** soportan **Ogg Theora** y **VP8**.
- **Chrome** soporta **Ogg Theora**, **H.264** (aunque puede que lo elimine en un fututo) y **VP8**.
- **Safari** soporta **H.264**.

Por lo tanto, el diseñador se encuentra en una situación en la que, si desea que sus vídeos sean compatibles con el mayor número de navegadores, tiene obligatoriamente que disponer de más de una versión del mismo archivo de vídeo.

Normalmente esto representa disponer de un archivo utilizando un códec libre, como **Ogg Theora** o **VP8**; y un archivo utilizando un códec propietario, como **H.264**.

Además, si se desea proporcionar compatibilidad también para navegadores antiguos, que no entienden la etiqueta **<video>**, tendrá que proporcionar el vídeo a través de **Flash**.

Veamos cómo lo indicaría todo esto en la etiqueta **<video>**:

```
<video width="350" height="280" controls
poster="media/poster.jpg">
<source src="media/coches.mp4" type="video/mp4">
<source src="media/coches.ogv" type="video/ogg">
<source src="media/coches.webm" type="video/webm">
Descárguese el vídeo desde
<a href="media/coches.mp4">aquí</a>.
</video>
```

Fíjate que ahora dentro de la etiqueta **<video>** se han incluido otras etiquetas adicionales **<source>** indicando, cada una de ellas, la ubicación de un archivo de vídeo.

Cada uno de estos archivos de vídeo es alternativo, por lo que el navegador del usuario utiliza únicamente el primero con el que es compatible.

La etiqueta <**source**> incluye el atributo **src** para indicar la ubicación del vídeo y el atributo **type** para indicar la naturaleza del mismo. Aunque este último atributo no es obligatorio, facilita el trabajo del navegador.

Además, se ha eliminado el atributo **src** de la etiqueta <**video**>, ya que ahora se emplean en su lugar varias etiquetas <**source**>.

Es conveniente escribir la etiqueta **<source>** correspondiente al formato **mp4** en primer lugar, ya que versiones antiguas del **iPhone** solo comprueban la primera etiqueta de este tipo.

Con esto solucionamos el problema para los navegadores compatibles con la etiqueta <**video**> de HTML5 pero todavía no para los navegadores antiguos.

La alternativa adecuada para estos casos es utilizar **Flash** si queremos que el vídeo se pueda reproducir directamente en el navegador o utilizar el código que proporcionan los sitios de hosting de vídeos, como **YouTube**.

Dicho código lo incluiremos dentro de la etiqueta <**video**>, como si fuera otra etiqueta <**source**>. Fíjate:

```
<video width="350" height="280" controls
 poster="media/poster.jpg">
<source src="media/coches.mp4" type="video/mp4">
<source src="media/coches.ogv" type="video/ogg">
<source src="media/coches.webm" type="video/webm">
<object type="application/x-shockwave-flash"
data="http://releases.flowplayer.org/swf/flowplayer-
3.2.1.swf" width="640" height="360">
   <param name="movie"
value="http://releases.flowplayer.org/swf/flowplayer-
3.2.1.swf" />
   <param name="allowFullScreen" value="true">
   <param name="wmode" value="transparent">
   <param name="flashVars"
value="config={'playlist':[{'url':'media%2Fcoches.mp4','
autoPlay':false}]}">
</object>
Descárguese el vídeo desde
<a href="media/coches.mp4">aquí</a>.
</video>
```

De esta forma hemos añadido el vídeo utilizando **Flash**, que utilizarán los navegadores antiguos. Hemos utilizado el mismo archivo **.mp4**, ya que Flash es compatible con el códec **H.264**.

Puedes obtener el código necesario en esta página web:

http://sandbox.thewikies.com/vfe-generator/

En resumen:

- Tendrás que codificar el vídeo en más de un formato.
- Al menos con un códec libre, como Ogg Theora o VP8.
- Al menos con un códec propietario, como H.264.
- Tendrás que utilizar Flash para los navegadores antiguos. Puedes aprovechar el mismo archivo MP4 (códec H.264) o webM (códec VP8).

4. CÓDECS DE AUDIO

Como se ha indicado, los archivos de vídeo normalmente también incluyen pistas de **audio** que utilizan códecs independientes de las pistas de vídeo.

Existe un gran número de códecs de audio, pero de cara a la Web podríamos decir que utilizaremos únicamente tres:

- **MP3** (códec propietario)
- **AAC** (códec propietario)
- **Vorbis** (códec libre)

Y como ocurre con el vídeo, los navegadores no implementan estos tres códecs, sino únicamente alguno de ellos.

Así pues actualmente tenemos que:

- **Internet Explorer** soporta los códecs **MP3** y **AAC**.
- **Firefox** y **Opera** soportan el códec **Vorbis**.
- **Chrome** soporta **Vorbis** y **AAC**.
- **Safari** soporta **MP3** y **AAC**.
- **Flash** soporta **MP3** y **AAC**.

De esta forma tenemos completa la "*suite*" de formatos y códecs que soportan cada navegador compatible con HTML5.

Habitualmente encontraremos las siguientes combinaciones en un archivo de vídeo:

- Formato **MPEG4** con códecs **H.264** (vídeo) y **AAC** (audio): es la versión con tecnologías propietarias.
- Formato **webM** con códecs **VP8** (vídeo) y **Vorbis** (audio): tecnologías libres.
- Formato **Ogg** con códecs **Theora** (vídeo) y **Vorbis** (audio): tecnologías libres.

Por ello, debemos indicar al navegador no solo el tipo de vídeo que hemos incluido, sino también los códecs utilizados.

Fíjate cómo lo haremos:

```
<source src="media/coches.mp4"
 type='video/mp4; codecs="avc1.42E01E, mp4a.40.2"'>

<source src="media/coches.ogv"
type='video/ogg; codecs="theora, vorbis"'>

<source src="media/coches.webm"
type='video/webm; codecs="vp8, vorbis"'>
```

Se indica tanto el formato de vídeo como los códecs utilizados. Primero el códec de vídeo y después el de audio.

Fíjate que como el valor del atributo **codecs** aparece entre comillas dobles, ha sido necesario utilizar comillas simples para **type**.

Al proporcionar esta información al navegador, este puede comprobar si es compatible no únicamente con el formato de vídeo o audio, sino también con los códecs utilizados.

5. LA ETIQUETA <AUDIO>

De forma similar podremos utilizar la etiqueta **<audio>** para reproducir un archivo únicamente de audio en una página web.

En este caso lo habitual es disponer de un archivo en formato propietario, como **MP3**; y otro en formato libre, como **Ogg**.

De esta forma abarcamos todos los principales navegadores y dispositivos móviles.

El formato **MP3** emplea el códec de su mismo nombre y el formato **Ogg** utilizará **Vorbis** como códec de audio. Además, para dar compatibilidad a navegadores antiguos, volveremos a utilizar **Flash**, ya que admite los códecs **MP3** y **ACC**.

Por lo tanto, el mismo archivo con el formato **MP3** podría servir para el objeto de **Flash**. Es lo mismo que ya hemos visto para el vídeo:

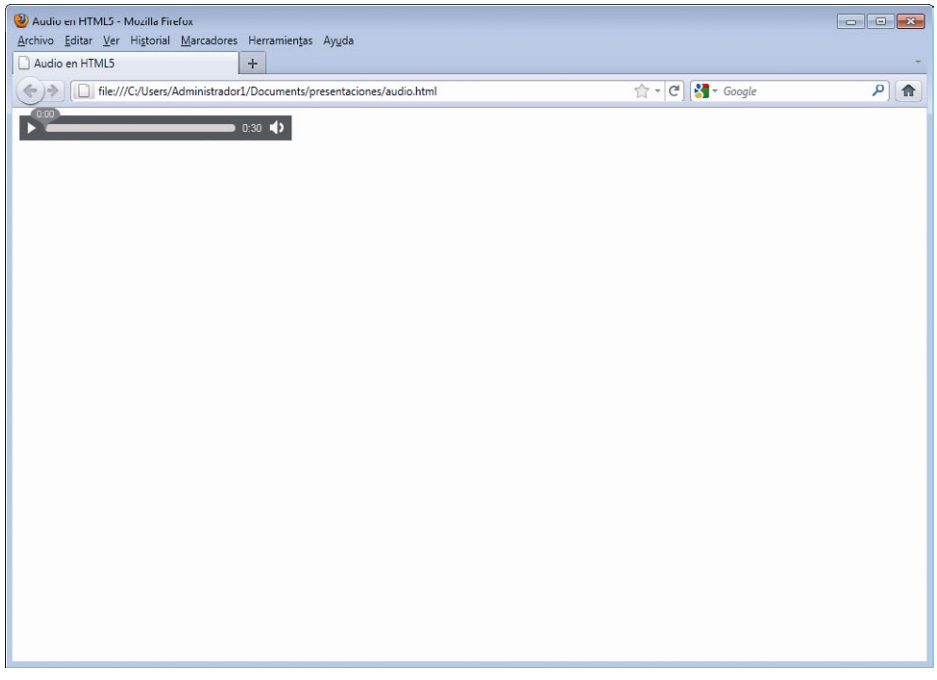
```
<audio controls>
<source src="media/Muestra.mp3" type="audio/mpeg">
<source src="media/Muestra.ogg"
 type="audio/ogg; codecs=vorbis">
</audio>
```

En este caso se usa la etiqueta **<audio>**, indicando con **<source>** los distintos archivos de audio alternativos.

Los navegadores que acepten el formato y códec **MP3** utilizarán el archivo **Muestra.mp3**; mientras que los compatibles con el formato **Ogg** y el códec **Vorbis**, utilizarán el archivo **Muestra.ogg**.

No he incluido el código correspondiente a la utilización de **Flash**, pero sería muy parecido al que hemos visto para el vídeo.

Los atributos de la etiqueta **<audio>** que pueden utilizarse son: **autoplay**, **controls**, **loop**, **preload** y **src**. Tienen el mismo significado que para la etiqueta **<video>**.



Existen varias aplicaciones para convertir un archivo de audio de un formato a otro. Te recomiendo realizar una búsqueda en Internet, ya que muchas de ellas son gratuitas.

6. API MULTIMEDA

Finalizamos la lección indicando que HTML5 no solo proporciona las etiquetas **<video>** y **<audio>** para incorporar elementos multimedia, sino que también facilita una **API** para manejar esos elementos mediante código.

Por ejemplo, esto podría servir para crear nuestro propio reproductor multimedia o conocer cuándo ha acabado de reproducirse el archivo.

La API para manejar los elementos multimedia consta de los métodos:

- **play**: para reproducir el archivo.
- **pause**: para detener la reproducción, pero el marcador de posición se queda en la situación actual. No hay un método "stop" para detener la reproducción y volver al principio del archivo.
- **load**: para empezar la carga del archivo multimedia.
- **canPlayType**: para comprobar si el navegador puede reproducir el archivo multimedia (es decir, si es compatible con el formato y códecs utilizados).

Además de varias propiedades y eventos.

Como ejemplo de cómo podríamos utilizar esta API, vamos a sustituir el atributo **controls** por dos botones que permitirán reproducir y detener el audio de esta página. Estudia el siguiente código:

```
<audio id="audio1" loop>
<source src="media/Muestra.mp3" type="audio/mpeg">
<source src="media/Muestra.ogg"
 type="audio/ogg; codecs=vorbis">
</audio>
<form id="frmControles" name="frmControles">
<input type="button" id="btn_play" value="Play">
<input type="button" id="btn_pause" value="Pause">
</form>
<script type="text/javascript">
var audio = document.getElementById("audio1");
```

```
document.getElementById("btn_play").onclick = function () {
   if (audio.paused || audio.ended)
   {
      if (audio.ended) {
       audio.currentTime = 0;
      }
      audio.play();
   }
}
document.getElementById("btn_pause").onclick = function ()
{
   if (!audio.paused) {
      audio.pause();
   }
}
</script>
```

Vemos que se ha eliminado el atributo **controls** de la etiqueta <**audio**>. Por lo tanto, no aparecerá el reproductor predeterminado que utiliza el navegador.

Además, se ha añadido un identificador para manejar este elemento en el código JavaScript y el atributo **loop**, para que se repita el audio indefinidamente.

Como no se visualiza el reproductor, utilizamos en su lugar dos botones de formulario con identificadores **btn_play** y **btn_pause**.

Seguidamente vemos el script para manejar la reproducción. En este caso es muy sencillo.

Se utiliza el método **getElementById** para obtener el elemento audio y reproducirlo (método **play**) o detenerlo (método **pause**).

Estos dos métodos se aplican al pulsar en los botones del formulario, por lo que están asociados al evento **click** de ambos.

Los métodos **play** y **pause** solo se ejecutan si tiene sentido hacerlo. Por eso se comprueba si el audio está en reproducción o no (propiedades **paused** y **ended**).

Puedes obtener más detalles acerca de esta API en la especificación de HTML5. Esto solo es un ejemplo muy sencillo para que veas cómo podrías utilizarla.

Además, esta API es exactamente la misma para el elemento de vídeo.