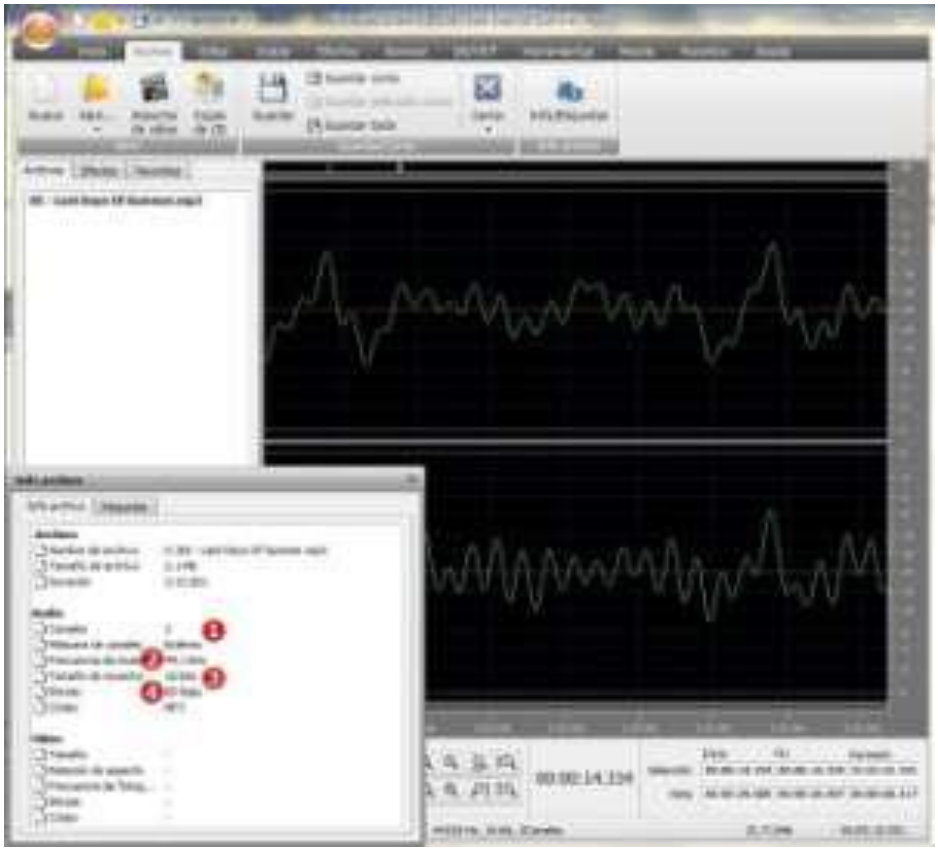
# Unidad 3. Implantación de contenidos multimedia

## 1. AUDIO: FORMATOS. CONVERSIONES DE FORMATOS (EXPORTAR E IMPORTAR)

Hay una serie de características de los archivos de sonido que es importante tener en cuenta a la hora de realizar una optimización de cara a su uso en la Web. Toda la información transmitida a través de Internet es digital, formada por cadenas de unos y ceros por lo que, la señal sonora, que es analógica, debe convertirse mediante un proceso de codificación (proceso mediante el cual, se transforma la señal sonora, que es analógica, en una señal digital formada por ceros y unos. Para ello se emplean los códecs de audio, que son algoritmos cuyo objetivo principal es reducir la cantidad de datos digitales necesarios para reproducir una señal auditiva) en el cual podremos poner en práctica nuestro conocimiento sobre las características de los archivos de sonido.



Entre las características de los archivos de sonido, cabe destacar las siguientes:

1. **Canales**: el valor 2 que muestra la ventana de información indica que nuestro archivo es Estéreo. De hecho, si observamos la parte oscura de la imagen, veremos que está dividida en dos zonas con ondas diferentes: la zona superior representa el canal izquierdo y la zona inferior representa el canal derecho. Otras opciones posibles respecto al número de canales serían: Mono o Sonido envolvente multicanal. numero de veces que se graba
2. **Frecuencia de muestreo**: el valor 44,1 KHz (kilohercios (un kilohercio es igual a mil muestras de una onda de sonido por segundo)) que muestra la ventana de información indica que nuestro archivo ha realizado 44.100 muestras por segundo de la onda de sonido analógica en el momento de realizar su codificación a digital, permitiendo registrar señales analógicas con componentes hasta los 20 kHz, con lo que se consigue un nivel de calidad de CD. Se conoce también como Tasa o Velocidad de muestreo. Este valor es proporcional a la calidad del sonido.

3. **Tamaño de la muestra**: el valor de 16 bits que muestra la ventana de información indica que nuestro archivo puede almacenar 65.536 valores de amplitud de ondas (distancia desde el punto más alto de la onda (desde el pico) hasta la base de la onda (el eje horizontal de equilibrio)) diferentes ofreciendo un rango de 96 dB (decibelios (unidad de medida utilizada para expresar el nivel de potencia e intensidad de un sonido)) y un nivel de calidad de CD. Este valor es proporcional a la calidad del sonido. También se conoce como Profundidad de bit o resolución de muestreo.
4. **Bitrate**: el valor de 80 kbps (kilobits por segundo) que muestra la ventana de información indica que nuestro archivo emplea 80.000 bits para almacenar cada segundo de sonido. Esta cifra varía en función de las tres anteriores y, sobre todo, del algoritmo empleado en su codificación. En el caso de la emisión de sonido en modo streaming (técnica que permite la reproducción de archivos multimedia sin necesidad de una descarga total previa del archivo que se reproduce a medida que se va descargando), su valor debe ser inferior a la tasa de bit de la conexión del usuario. También se conoce como Tasa de bit.

Hay que tener en cuenta que todos estos valores influyen proporcionalmente en el peso del archivo, por lo que, disminuyendo cualquiera de ellos disminuirá su peso, lo cual resultará beneficioso para el usuario.

No hay que olvidar que, con características iguales, será la duración de un archivo de sonido la que influya en su peso, por lo que, a la hora de incorporar un sonido a nuestra Web habrá que preguntarse si su duración es la necesaria o podríamos conseguir el efecto deseado en la audiencia con una duración menor, en cuyo caso y en beneficio de esa audiencia, habrá que recortar el archivo para que ocupe lo mínimo imprescindible.

## 1.1. FORMATOS

Al hablar del audio en Internet, el formato más conocido es **mp3**. Éste es un formato de compresión de audio digital patentado que usa un **algoritmo con pérdida** para conseguir un menor tamaño de archivo. Es un formato de audio común usado para música tanto en ordenadores como en reproductores de audio portátil. Más concretamente, mp3 fue desarrollado por el Moving Picture Experts Group (MPEG) para formar parte del estándar MPEG-1 y del posterior y más extendido MPEG-2. Un mp3 puede comprimirse usando una mayor o menor tasa de bits por segundo, resultando directamente en su mayor o menor calidad de audio final, así como en el tamaño del archivo resultante.

Es importante destacar que, en lo que respecta al desarrollo web y al uso del audio para ser interpretado por los diferentes navegadores, mp3 no es único y universal, es decir, hay más formatos iguales o mejores y **no todos los navegadores lo soportan** (por ejemplo, Firefox no interpreta mp3 por sí solo). Por lo tanto, hay otras posibilidades de formatos que hay que conocer, sobre todo para, como luego veremos, permitir que la reproducción de audio sea lo más universal posible entre todos los navegadores actuales (al margen de sus batallas empresariales).

Cuando se habla de archivos comprimidos es importante saber distinguir entre formato y codec. El formato es la estructura del archivo que guarda el audio, mientras que códec hace referencia al tipo de compresión que se utiliza o ha utilizado para comprimir un audio en un archivo. Comparado con la imágenes, el formato es cómo se guarda la imagen (jpg, tiff, bmp, etc.) y el códec es lo equivalente a usar una compresión rar o zip para esos archivos. Para reproducir un audio en un

formato que es comprimido (codificado) con un códec determinado se necesita ese códec para decodificarlo.

Los siguientes formatos no son los únicos que existen, pero sí son los más conocidos en Internet, y los más usados en la distribución de audio en sitios web.

- **Ogg**: es un formato válido para audio y vídeo. Es un formato contenedor, desarrollado por la Fundación Xiph.org. Su principal ventaja con respecto a mp3 es que es un formato libre de patentes y abierto diseñado para dar un alto grado de eficiencia en el streaming y la compresión de archivos. De manera incorrecta, muchas veces los archivos ogg se les llama Vorbis ya que éste fue el primer códec desarrollado para leer este formato. Sin embargo, hay muchos otros códec además de Vorbis que interpretan ogg. Actualmente, Firefox y Chrome interpretan ogg.
- **Real Audio**: sin duda, este formato es el rey del streaming. La gran mayoría de las páginas que usan streaming utilizan el formato Real Audio para difundir sus contenidos. RealNetworks es la compañía que ha creado Real Audio y, por supuesto, también ha creado un reproductor (player en inglés) para su formato, el RealPlayer (del que existe una versión gratuita y otra comercial).
- **Windows Media Audio (wma)**: es un formato desarrollado por Microsoft. Ofrece gran calidad de sonido en un tamaño reducido. Y como ventaja para las compañías y los músicos, el formato de Microsoft incluye un sistema de gestión de los derechos de autor para evitar la piratería y la creación de copias no autorizadas de las canciones. Por las características de calidad y tamaño wma resulta muy adecuado para la reproducción de archivos en streaming. El reproductor que Microsoft ha desarrollado para escuchar su formato de audio es el Windows Media Player que se encuentra disponible en su web. La desventaja de wma es que, al ser de Microsoft, el navegador lExplorer lo soporta, pero no se puede asegurar en el resto.
- **wav** o wave (WAVEform audio file format): es un formato de audio que se suele utilizar sin compresión de datos, wav es propiedad de Microsoft y de IBM. Este formato es muy conocido en entornos Windows, pero no se suele usar en Internet por ocupar mucho espacio al ser usado sin compresión.
- **vqf**: el formato de audio vqf ha sido desarrollado por la compañía Yamaha y presenta algunas ventajas respecto al mp3: mayor calidad de sonido y archivos de menor tamaño (se puede comprimir hasta un 30% más que mp3). El reproductor más asociado a este archivo es el Yamaha SoundVQ, aunque hay más que lo soportan. El mayor inconveniente del vqf es que no está muy extendido (por lo que muchos navegadores no lo soportan), además, su reproducción necesita de más recursos que cualquiera de los formatos anteriores.

## 1.2 CONVERSIÓN DE FORMATOS

Un hándicap importante a la hora de trabajar con audio en páginas web es saber cómo implementa cada navegador los reproductores. No todos los navegadores tienen reproductores para todos los formatos, sino que hay preferencias y lo que un navegador puede reproducir otro no. Esta situación lo único que hace es complicar el trabajo del desarrollador, ya que éste debe intentar garantizar que los audios que inserte en un sitio web puedan ser reproducidos por todos los navegadores, y si eso no es posible, por aquellos más utilizados por el usuario objetivo del portal.

La Tabla muestra una comparativa de qué formatos son reproducidos por los 5 navegadores más destacados actualmente (bajo Windows). Esta lista no es definitiva, ya que la vertiginosa actualización de los navegadores hace que las prestaciones que ofrecen en una versión sean mejoradas o reducidas en las siguientes versiones.

Formatos de audio y navegadores

| Formato | IE | Firefox | Opera | Chrome | Safari |
|---|---|---|---|---|---|
| **Ogg** | No | Sí | Sí | Sí | No |
| **Mp3** | Sí | No | No | Sí | Sí |
| **Wav** | No | Sí | Sí | Sí | Sí |

Los valores de Tabla hay que tenerlos en cuenta a la hora de saber qué formatos insertar como audio en un sitio web. Como se verá en los siguientes puntos, HTML5 ofrece una alternativa para insertar varios formatos de un audio dependiendo de cuál es el que soporta un determinado navegador. El desarrollador guarda el mismo audio en diferentes formatos, y en HTML5 se referencian todos ellos para que cada navegador elija el que puede reproducir.

Para poder tener un audio en varios formatos, es necesario usar herramientas de conversión. Estas herramientas suelen soportar una gran cantidad de formatos y suelen ser sensibles respecto a los parámetros de compresión en aquellos que lo soportan.

Herramientas de conversión hay muchas disponibles en Internet. Algunas gratuitas son:

- **Free Studio** (http://www.dvdvideosoft.com/es/free-dvd-video-software.htm): es una potente herramienta para convertir todo tipo de archivos (vídeo, audio, imagen). Es fácil de usar y muy adecuada para un diseñador que trabaja con audio en la web. Solo versión para Windows.
- **Audacity** (http://audacity.sourceforge.net/?lang=es): es una herramienta avanzada para grabar y editar. Es libre y de código abierto. Su funcionalidad es mucha, y entre ella está la de convertir archivos. Es muy adecuada si el diseñador desea crear sus propios sonidos. Para Linux y Windows.

También hay conversores online, servicios web a los que se le sube el archivo de audio, se selecciona el formato al que se quiere convertir y el resultado se envía por email. Un ejemplo de este tipo de herramientas es **Switchr** (http://switchr.net), sin embargo hay muchas más disponibles en Internet.

Para poder probar estas herramientas y hacer las prácticas necesitarás archivos de música de muestra, en este artículo tienes varias páginas donde poder encontrar música libre de derechos:

https://aulacm.com/paginas-descargar-musica-libre-derechos/

## 1.3. INSERTAR AUDIO EN UNA WEB

Vistos los diferentes formatos de audio adecuados para Internet, cuáles son soportados por los navegadores más adecuados, y herramientas de conversión de formatos, en este punto se muestran diferentes maneras de introducir audio en una web con HTML.

### 1.3.1. AUDIO DE FONDO (EN HTML4)

Poner un sonido de fondo para una página web usando etiquetas HTML no es la opción más versátil. Existen dos etiquetas que, en teoría, lo permiten <bgsound> y <embed>.

DEPRECATED

**<bgsound>** es una alternativa aparentemente adecuada pero con el inconveniente de que es solo entendible por Internet Explorer. Evidentemente, esto la hace muy poco portable por lo que no se recomienda su uso, salvo en situaciones muy concretas. Un ejemplo de uso para ejecutar de fondo un sonido audiosIBeethoven.wma (solo válida para lExplorer):

```
<bgsound  src="audios/Beethoven.wma"   loop="2" volume="20"   />
```

Lo más importante de esta etiqueta es que para indicar el fichero a reproducir hay que utilizar el atributo **src**. Otros atributos de interés son **loop**, que permite que la canción vuelva a empezar cada vez que acaba (se le puede indicar cuántas veces se repetirá, con INFINITE no hay límite); **delay**: es un valor para retrasar el inicio de la música (debe ser un número positivo en segundos); volumen, determina la intensidad del sonido de fondo, con valores entre -10.000 (el más débil) a O (el más alto).

**<embed>** es una alternativa para insertar complementos (plugins) de audio y vídeo que es compatible para todos los navegadores. Sin embargo, no está incluida dentro del estándar W3C por lo que su implementación depende de los navegadores, y no todos las interpretan igual. Además, dependiendo del formato del archivo, así lo interpretarán los navegadores. Por ejemplo, en Google Chrome no funciona el reproductor por defecto cuando la extensión es wma (Microsoft) pero sí con mp3. Un ejemplo de código es el siguiente:

```
<embed src="audios/Aretha.mp3" height="0" type="audio/mpeg" loop="false" controller="false">
```

El atributo **src** sirve para especificar la ruta del archivo (como en <bgsound>). Los audios insertados a través de esta etiqueta se reproducen automáticamente al cargarse la página, y se reproducen solamente una vez. Esto puede cambiarse a través de los atributos autostart y loop. El atributo **autostart** indica si el archivo se reproducirá automáticamente al cargarse la página, y puede tomar los valores true o false (aunque no funciona correctamente para todos los navegadores) El atributo **loop** indica si el archivo se reproducirá continuamente en un bucle, y también puede tomar los valores true o false. Los atributos **width** y **height** sirven para especificar el tamaño de la consola de control de sonido (o vídeo). Estos atributos pueden tomar como valor un número, que indica el tamaño en píxeles. Si no se especifican estos atributos, la consola de control de vídeo se mostrará con el tamaño más adecuado al tamaño del sonido. **Controller** indica si aparece o no la consola de controles del reproductor. **Type** es un atributo importante, que declara el tipo de fichero de audio que estamos usando, con lo que el navegador web puede ejecutar el complemento o plugin adecuado para la reproducción del fichero, Puede ser audio/midi, audio/wav o audio/mnpeg, entre otros.

En cualquier caso, porque depende del navegador y del plugin necesario para su inclusión, ninguna de estas dos alternativas ofrece una solución interoperable entre todos los navegadores, por tanto, es poco recomendable su uso.

### 1.3.2. DESCARGAR AUDIO POR EL USUARIO (EN HTML4)

La manera más sencilla de incluir sonidos es dejando al usuario la decisión de escucharlos o no en local. En realidad es más una opción descargar que reproducir. Para ello se puede colocar el nombre del archivo de sonido en una atributo href de la etiqueta <a>. El siguiente código muestra un ejemplo.

```
<a href="audios/Beethoven.wma">Audio de Beethoven<a>
```

En este ejemplo, dependiendo de la versión del navegador que se esté usando, se le da al usuario la opción de descargar el archivo en local o se carga un complemento (plugin) que lo ejecuta. Pero todo necesita de descargarse en local y ejecutarlo con un reproductor (player).

A USAR

### 1.3.3. INSERTAR SONIDO (HTML5)

En HTML5 la inclusión de sonidos en páginas web intenta paliar muchos de los inconvenientes vistos con las etiquetas <bgsound> o <embed>. De alguna manera, HTML5 ofrece una alternativa más sencilla y eficaz para que el desarrollador pueda insertar audio. Sin embargo, la realidad es que cada navegador implementa de HTML5 lo que le quiere, por lo que tampoco con esta opción se consigue una solución universal.

La propuesta de HTML5 utiliza la etiqueta **<audio>.** Esta etiqueta es nueva en HTML5 y no existe para versiones de HTML4. Evidentemente, eso obliga a que los navegadores que la interpreten deban implementar HTML5 en general (y esta etiqueta en particular).

Los atributos de esta etiqueta son muy parecidos a los vistos para <embed>.

**Autoplay** especifica que el audio debería empezar automáticamente tan pronto como esté listo para reproducirse (el único valor de este atributo es autoplay). **Controls** indica que los controles de reproducción serán visibles (el único valor posible de este atributo es controls). **Loop** indica que el audio se repetirá automáticamente cuando termine (el único valor posible de este atributo es loop). **Preload**, especifica sí el audio debería ser cargado cuando la página se carga o no, es decir, se empieza a cargar cuando el usuario lo quiera reproducir (el valor *auto* indica que se cargará todo el archivo cuando se cargue la página, el valor *meta* indica que solo se cargarán los metadatos del archivo y *none* que no habrá precarga). Por último, **src** especifica la URL de archivo de audio. Además de estos atributos, <audio> soporta también atributos globales de HTML5 y eventos.

Un ejemplo del uso de esta etiqueta para obtener un reproductor para un audio que se inicia solo, sería:

```
<audio src="audios/Aretha.mp3" controls="controls" autoplay></audio>
```

Este ejemplo funciona correctamente en IExplorer y en Google Chrome . La apariencia de los reproductores es diferente, pero ambos reproducen el archivo mp3 obedeciendo al atributo autoplay. Es importante destacar que el uso de HTML5 abre las puertas de navegadores de dispositivos móviles como IPad, IPhone y los basados en Android o Windows. El caso de IPad (que hasta la fecha) no interpreta archivos Flash, la etiqueta <audio> ofrece una alternativa muy elegante y sencilla para los desarrolladores.

Una mejora para hacer más versátil esta opción es usar la etiqueta <**source**> dentro de <audio>. De esta manera, el navegador intenta cargar la primera línea <source>. Si falla o no es soportada esa reproducción, entonces pasa a la siguiente. Por ejemplo, si IExplorer, Safari y Google Chrome leen archivos mp3 y Firefox lee archivos ogg (no soporta mp3), el siguiente código hace una solución para los cuatro navegadores. Cuando el navegador Firefox llegue al primer <source> y no

pueda interpretar mp3 entonces pasará al segundo con ogg. Sin embargo, Internet Explorer, Safari y Google Chrome se quedarán con el primer <source>. El siguiente ejemplo empieza definiendo el tipo de documento (<!DOCTYPE html> )que es: navegadores como lExplorer lo necesitan para interpretar bien el contenido.

```
<!DOCTYPE html>
<html>
  <head>
  </head>
   <body>

      <audio controls="controls" preload="auto" >
      <source src="Aretha.mp3" type="audio/mpeg" />   distintos formatos para el mismo audio
      <source src="Aretha.ogg" type="audio/ogg" />
      El navegador NO soporta el audio </audio>
   </body>
</html>
```


Reproductor en Google Chrome con etiqueta <audio>

En resumen, este código funcionará en los navegadores más conocidos: lExplorer, Firefox, Google Chrome y Safari pero no funciona en IExplorer8. Si se quitara la línea con el archivo ogg entonces Firefox 7 no lo soportaría ya que no lee archivos mp3, pero sí en el resto de navegadores antes enumerados. Por otro lado, si se quitara la línea con el archivo mp3, en Google Chrome 14 y Firefox 7 funcionaría (porque leen .ogg), pero en lExplorer 9 y Safari 5 no.

A modo de conclusión, el uso de la etiqueta <audio> hace que insertar audio en la web sea bastante interoperable entre diferentes navegadores (aquellos que interpretan HTML5). Sin embargo, un diseñador, debería hacer soluciones que también la soporten otros navegadores más antiguos. Es decir, en el ejemplo anterior ¿qué pasa si un usuario tiene IExplorer8? Si ese usuario no puede ver correctamente los elementos de su sitio web lo más seguro es que lo abandone defraudado.

## USO DE <OBJECT> COMO ALTERNATIVA

Una alternativa incluida dentro de HTML4 para insertar objetos no soportados por el navegador es el uso de la etiqueta **<object>**. Combinando <object> con <embed> se puede, por ejemplo, insertar un reproductor (Flash) para aquellos navegadores que no soporten el código mostrado en el ejemplo anterior (en nuestro caso solo era lExplorer8). Esto ofrece una solución más interoperable, aunque tampoco es universal.

El siguiente código muestra un ejemplo de uso de <object> con <audio>. Si ninguna de las opciones ofrecidas en <source> es soportada por el navegador entonces intentará embeber el reproductor

player_mp3_mini.swf (éste no es el único reproductor Flash, hay muchos más disponibles en Internet).

```
<!DOCTYPE html>
  <html>
  <head>
  </head>
  <body>
      <audio controls="controls" preload="auto">
      <source src="Aretha.mp3" type="audio/mpeg"/>
      <source src="Aretha.ogg" type="audio/ogg"/>
      <object type="application/x-shockwave-flash"
data="player_mp3_mini.swf" width="200" height="20">
     <param name="movie"  value="player_mp3_mini.swf"/>
     <param name="bgcolor"  value="#085c68"   />
     <param name="FlashVars"  value="mp3=Aretha.mp3"   />
     <embed href="player_mp3_mini.swf" bgcolor="#085c68" width="200" height="20"
name="movie"   align="" type="application/x-shockwave-flash"
flashvars="mp3=Aretha.mp3">
     </embed>
    </object>
    </audio>
  </body>
</html>
```

Para que este código funcione es necesario que el archivo player_mp3_mini.swf esté disponible en la misma carpeta (local o en web) que la página HTML. Este archivo es el que carga <object> cuando ninguna de las otras opciones son soportadas. <object> tiene varios parámetros que definen el tamaño del elemento que quiere embeber, en este caso el reproductor Flash. Sin embargo, es con la etiqueta <embed> con la que realmente se está incluyendo el archivo mp3 que se quiere cargar en ese reproductor. De alguna manera, con este código se pretende insertar un audio mp3 con la etiqueta <embed> vistan anteriormente, pero añadiendo adrede el reproductor, y no suponer que el navegador lo tiene.

La etiqueta <object> es muy conocida en HTML4 por lo que no es objetivo de este libro tratarla en profundidad. Sin embargo, si es interesante resaltar que los valores de los atributos coincidentes entre <object> y <embed> deben tener los mismos valores para garantizar un funcionamiento adecuado.

### APLICAR ESTILOS A AUDIO

A la etiqueta <audio> también se le puede aplicar estilos CSS. Por ejemplo, el siguiente código define un estilo para hacer que una etiqueta audio que use la clase audio-fondo aparezca con una color de fondo amarillo y una tamaño determinado.

```
<style>
      .audio-fondo {
      width: 160px;
      height: 36px;
      background: #FFFFOO; }
</style>
```

A modo de conclusión de lo visto, aunque no todos los navegadores actuales soportan HTML5 con la madurez deseada, y no todos los usuarios usan los navegadores debidamente actualizados para garantizar que soportan HTML5, la etiqueta <audio> ofrece muchas ventajas que aconseja su uso futuro a la hora de insertar audio en sitios web. HTML5 permite que los navegadores yo no necesiten complementos (plugins), del estilo Adobe Flash, Microsoft Silverlight, para reproducir audio y eso le facilita el desarrollador no tener que preocuparse si un navegador reproduce o no un audio determinado.

Los dispositivos móviles actuales integran navegadores que soportan HTML5 por lo que es cuestión de poco tiempo que todos los usuarios tengan navegadores HTML5 y no sea necesario buscar alternativas con <object> o JavaScript para detectar que audio puede reproducir un complemento determinado de un navegador concreto. Aunque será inevitable que en la etiqueta <audio> se den opciones variadas respecto al tipo de archivo (mp3, ogg).

### 1.3.4. LA API DE AUDIO

Este sería, en resumen, el API de JavaScript para audio, similar al que veremos más adelante para el vídeo.

| Métodos | Propiedades | Eventos |
|---|---|---|
| **play()** | currentSrc | Play |
| **pause ( )** | currentTime | Pause |
| **load()** | startTime (readonly) | Progress |
| **canPlayType ( )** | videoWidth | Error |
| | videoHeight | Timeupdate |
| | duration (readonly) | Ended |
| | ended (readonly) | Abort |
| | Error | Empty |
| | paused (readonly) | Emptied |
| | Muted | Waiting |
| | seeking | loadmetadata |
| | volume desde 0 a 1, para cambiar sumar o restar 0.1 | |
| | Height | |

| Width |
| --- |
| seekable (readonly) |
| played (readonly) |

Como opción de diseño avanzada, se puede usar JavaScript y HTML5 para personalizar el reproductor. El siguiente código utiliza botones para reproducir, en vez de un player. Esta opción da mucho juego en el diseño del reproductor al poder cambiar los botones por imágenes acordes con el estilo del sitio.

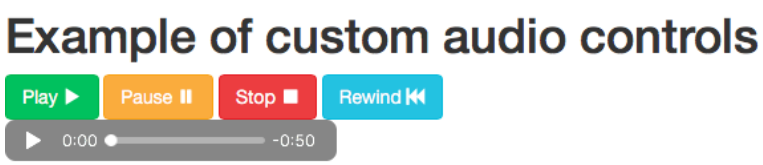
```
<!DOCTYPE html>
<html>
<head>
</head>
<body>
<audio id="player" src="Aretha.mp3"></audio>
<div>
<button onclick="document.getElementById('player').play()">Play</button>
<button onclick="document.getElementById('player').pause()">Pausa
</button>
<button onclick="document.getElementById('player').volume+=0.1">
+ Volumen</button>
<button onclick="document.getElementById('player').volume-=0.1">
- Volumen</button>
</body>
</html>
```

A continuación, tienes un ejemplo un poco más elaborado: cargar el audio en una variable de js y aplicarle los disntintos metodos y propiedades

```
<!DOCTYPE HTML>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>Simple audio control</title>
  <link rel="stylesheet" href="css/bootstrap.min.css" type="text/css">
</head>
<body>
    <header>
        <h1>Example of custom audio controls</h1>
        <button onclick="playAudio();" class="btn btn-success">
            Play <i class="glyphicon glyphicon-play"></i>
        </button>
```

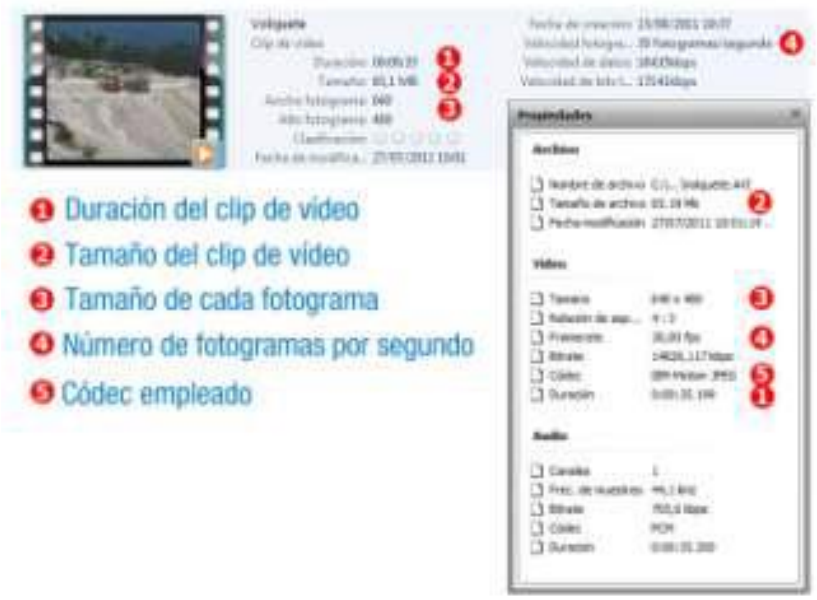
```
        <button onclick="pauseAudio();" class="btn btn-warning">
            Pause <i class="glyphicon glyphicon-pause"></i>
        </button>
        <button onclick="stopAudio()" class="btn btn-danger">
            Stop <i class="glyphicon glyphicon-stop"></i>
        </button>
        <button onclick="rewindAudio();"  class="btn btn-info">
            Rewind <i class="glyphicon glyphicon-fast-backward"></i>
        </button>
       </header>
<audio controls="controls" id="myAudio">
   <source src="audio_clip1.mp3" type="audio/mp3" />
   <source src="audio_clip1.ogg" type="audio/ogg" />
   Your browser does not support the <audio> element.
</audio>

<script>
    theAudio = document.querySelector("#myAudio");

    function playAudio() {
       theAudio.play();
    }

    function pauseAudio() {
       theAudio.pause();
    }

    function stopAudio() {
       theAudio.currentTime = 0;
       theAudio.pause();
    }

    function rewindAudio() {
       theAudio.currentTime = 0;
    }
</script>
</body>
</html>
```

Aspecto:




## 2. VÍDEO: CODIFICACIÓN DE VÍDEO, CONVERSIONES DE FORMATOS (EXPORTAR E IMPORTAR)

El vídeo digital es un tipo de sistema de grabación de vídeo que funciona usando una representación digital de la señal de vídeo, en vez de una representación analógica. El vídeo puede obtenerse por grabación directa con una cámara de vídeo digital o por la digitalización de un vídeo analógico. Toda la información que se transmite a través de Internet es digital, es decir, mediante cadenas de unos y ceros.




Hay una serie de características de los archivos de vídeo que es importante tener en cuenta a la hora de realizar una optimización de cara a su uso en la Web. Estas características son:

1. **Duración del clip** (película o vídeo de corta duración): tiempo que dura la reproducción de vídeo. Se presenta en el formato HH:MM:SS.
2. **Tamaño del clip**: es el espacio que ocupa el vídeo en el dispositivo de almacenamiento. Se mide normalmente en MB.
3. **Tamaño de cada fotograma** (cada una de las imágenes individuales captadas por cámaras de video y registradas analógica o digitalmente): es el tamaño en píxeles de cada fotograma distinguiendo el ancho y el alto. De este tamaño se desprende otra característica, la **Relación de aspecto**, que es la comparación del ancho respecto al alto. Los valores habituales para la relación de aspecto son 4 : 3 (normal) y 16 : 9 (panorámico).
4. **Número de fotogramas por segundo (fps):** es el número de imágenes estáticas que se suceden en un segundo y que al visionarse secuencialmente transmiten la sensación de movimiento.

5. **Códec**: algoritmo de compresión/descompresión usado durante la generación del archivo.


| NÚMERO | DESCRIPCIÓN | ANTES | DESPUÉS |
|---|---|---|---|
| 1 | Peso del archivo | 65,1 MB | 2,51 MB |
| 2 | Tamaño del fotograma (Ancho x Alto) | 640 x 480 | 320 x 240 |
| 3 | Tasa de fotogramas | 30 fps | 25 fps |
| 4 | Formato original | AVI | |
| 5 | Tipo de conversión | | WMV |
| 6 | Perfil de la conversión | | Calidad normal - 568 Kbps |

La imagen que ilustra el apartado es una composición de dos capturas de pantalla: la primera, colocada en la parte superior de la imagen, se corresponde con la información mostrada por el explorador de archivos del Windows 7 cuando el archivo está seleccionado y la segunda, colocada en la parte inferior derecha, presenta el cuadro de diálogo de propiedades del archivo que sale cuando éste forma parte de la biblioteca media (nombre que da el programa AVS Video Editor al lugar donde se muestran todos los archivos multimedia importados para trabajar en un proyecto de edición de vídeo) del programa AVS Video Editor. En la imagen se encuentran señaladas, con los números 1, 2, 3, 4 y 5 en letra blanca sobre un círculo rojo, las características mencionadas.

Cuando incorporamos vídeo a la Web debemos recordar que no todos los usuarios tienen una buena conexión por lo que, si es un vídeo para descarga completa y posterior reproducción, debemos tener en cuenta el peso del archivo, tras que, en el caso de utilizar una solución de **streaming**, debemos tener en cuenta la tasa de transferencia o tasa de bit, es decir, el número de

bits por segundo a la que se debe transmitir el vídeo para que el usuario lo pueda reproducir sin interrupciones.

Para añadir un vídeo a nuestra web para descarga y posterior reproducción lo hacemos de la misma forma que para añadir un sonido. Creamos un enlace al clip utilizando la etiqueta del HTML correspondiente al elemento ancla, tal y como se puede ver en la parte superior de la imagen que ilustra este apartado. En este caso, se debe informar, de alguna manera, al usuario de las características del archivo de forma que pueda decidir si desea o no pinchar en el enlace. La parte inferior de la imagen que acompaña a este párrafo muestra cómo se vería en el navegador el código XHTML mostrado en la parte superior de la imagen y, en la esquina inferior derecha, apuntado por una flecha roja, se puede ver el reproductor de vídeo que veríamos al pulsar sobre el enlace con los botones para parar, retroceder, pausar y avanzar el clip.

También y de forma similar a como hacíamos con el sonido, podemos incrustar en la propia página un reproductor de vídeo como si se tratara de una imagen. El usuario tendrá que tomar la decisión de visionar el vídeo o no.

## 2.1. FORMATOS DE VÍDEO

Mucho de lo visto en la sección anterior respecto a los formatos de audio es aplicable a vídeo. El vídeo tiene por un lado el formato de los archivos (avi, mpeg, etc.) y por otro los codec con los que se comprime (codifica) y descomprimen (decodifica) y que puede haber varios para un mismo formato. Este proceso de compresión tiene más sentido en vídeo que en audio ya que no es posible difundir vídeo por Internet si éste no está comprimido (debido al espacio que ocuparía).

A continuación, se hace un resumen de los formatos más conocidos en Internet y los codec asociados a ellos. Estos formatos son en realidad formatos contenedores, es decir, un tipo de formato de archivo que almacena información de vídeo, audio, subtítulos, capítulos, meta-datos e información de sincronización siguiendo un formato preestablecido en su especificación:

- .**mp4**: es un formato que puede contener vídeo en formato MPEG-4, es decir el creado con los codee: DivX, Xvid, QuickTime y H.264, entre otros.
- **.swf, .flvy .f4v**: son los sucesivos formatos que Adobe ha ido definiendo para el Flash vídeo desde 2002 hasta ahora. Las últimas versiones soportan los códec Sorenson Spark, VP6 y H.264. El .f4v solo soporta H.264.
- **.ogg** y **.ogv**: ogv es el correspondiente contenedor Open Source de la Fundación Xiph.Org. Apropiado para contener el formato Theora.
- **.mkv** (Matroska): es un formato Open Source que puede contener casi cualquier tipo de formato de vídeo. Muy usado originalmente para comprimir películas que se han de compartir por Internet.
- **.webm** (WebM): es un contenedor de vídeo Open Source desarrollado por Google, muy dirigido para usarse con HTML5. Está compuesto por el codec VP8 y el codee de audio Vorbis (ogg) dentro de un contenedor multimedia Matroska.
- **.avi**: formato contenedor propietario de Microsoft. Además de los formatos de Microsoft soporta otros muchos, entre ellos MPEG-4. No tiene un uso muy extendido en páginas web.

- **.mov**: es el fichero contenedor de QuickTime propietario de Apple. En realidad es casi idéntico a .mp4, ya que el MPEG se basó en QuickTime para definirlo; Pero al ser propietario de Apple tiene menos soporte en otras plataformas.

## 2.2. CONVERSIÓN DE FORMATOS

Como ocurre con el audio, existen muchas aplicaciones que permiten convertir de un formato de vídeo. En el desarrollo web, estas aplicaciones ofrecen una alternativa sencilla para poder dar soluciones flexibles para la mayor cantidad posible de navegadores.

Al igual que con el audio, existe una "batalla" entre fabricantes de navegadores sobre que formato de vídeo debe de ser el estándar y por supuesto no todos reproducen los mismos formatos de forma nativa. Al usar HTML5 ocurre igual que con los ficheros de audio, unos navegadores contemplan de manera nativa unos códec y otros apuestan por otros, y las diferencias son irreconciliables. La siguiente tabla muestra los códec en los principales navegadores:

Formatos de vídeo y navegadores

| Códec | Tipo | IE | Firefox | Chrome | Safari | Opera |
|---|---|---|---|---|---|---|
| **Ogg Theora** | Libre | No | Sí | Sí | No | Sí |
| **H.264** | Propietario | Sí | No | No | Sí | No |
| **VP8** | Libre | No | Sí | Sí | No | Sí |

Los formatos más usados con HTML5 (y en los ejemplos siguientes) usan los siguientes códec para vídeo y audio:

mp4 = H.264 (vídeo) +AAC (audio)

ogg/ogv = Theora (vídeo) + Vorbis (audio)

webm = VP8 (vídeo) + Vorbis (audio)

De esta tabla, se puede interpretar que, por ejemplo, ogv es un formato adecuado para Firefox, pero que IEplorer no lo soporta, y le va mejor un formato mp4 como ha Safari.

Herramientas de conversión hay muchas disponibles en Internet. Algunas gratuitas son:

- **Miro Video Converter (http://www.mirovideoconverter.com/)**: es una utilidad muy sencilla para convertir a cualquier formato de vídeo. Incluye ogv y webm.
- **Free Studio (http://www.dvdvideosoft.com/es/free-dvd-video-software.htm)**: es una potente herramienta para convertir todo tipo de archivos (vídeo, audio, imagen). Es fácil de usar y muy adecuada para un diseñador que trabaja con audio y vídeo en la web. Solo versión para Windows. La versión 5.2.1 y anteriores no soporta conversión a ogv.
- **AtubeCatcher (http://atube-catcher.dsnetwb.com):** es una utilidad muy interesante. Además de convertir archivos a una gran cantidad de formatos, permite descargar (en varios formatos) vídeos de los repositorios más conocidos (Youtube, Google Videos o Vaneo).

## 2.3. EDICIÓN Y MONTAJE DE VÍDEOS.

Los primeros pasos para realizar un vídeo para la Web coinciden con los pasos para realizar cualquier vídeo:

- **Planificación**: Tenemos que disponer de un guion previo de lo que queremos hacer, que dependerá de lo que queramos transmitir. Deberemos filmar las escenas o recurrir a escenas que tengamos ya filmadas. En este paso debemos decidir la duración total del vídeo, ya que de ello dependerán los recortes o añadidos que tengamos que hacer a las secuencias que tengamos filmadas. Debemos saber si vamos a incorporar imágenes para separar las secuencias filmadas, si nos interesa el audio de la grabación o lo vamos a sustituir por otro o si lo vamos a subtitular. Todo es importante.
- **Edición y montaje**: En este paso, emplearemos un software apropiado que nos permita realizar todas las tareas previstas en la planificación.

En la imagen que acompaña a este párrafo se puede ver una instantánea del proceso de edición y montaje de un vídeo realizada con el programa AVS vídeo Editor en la cual se identifican algunas de las zonas más interesantes.


Hoy en día es muhy normal hacer este trabajo utilizando herramientas online, que nos ofrecen gran variedad de plantillas de vídeos y presentaciones cortas, ejemplos, fuentes, sonidos, etc. Tiene varios ejemplos en el siguiente artículo https://elidiomadelaweb.com/10-sitios-web-para-crear-videos-y-animaciones-de-forma-sencilla/.

En esta infografía puedes encontrar otras herramientas para distintas actividades con vídeos:


Como curiosidad la aplicación Widnows Movie Maker de Microsoft que se ha usado con anterioridad a Windows 10 ya no está disponible, afreciando una servicio semejante en la aplicación Fotos, que permite realizar montajes de fotos y videos, añadiendo pistas de música, títulos, efectos, efectos 3D

## 2.4. EXPORTACIÓN DE VÍDEOS.

Ya hemos visto en apartados anteriores que hay muchos formatos que contienen audio, vídeo, subtítulos, capítulos y que pueden tener varias pistas de audio o vídeo.

Puede que en algún momento interese tener el vídeo almacenado en un archivo de menor tamaño sin estar acompañado del resto de información.

También puede que interese poder ver ese vídeo en otro dispositivo como un móvil o una consola. Aunque esta operación podría ser, también, una simple conversión de formato como es el caso de la imagen que acompaña a este párrafo en el cual se ha empleado el programa **AVS Video Converter** para exportar un vídeo en formato AVI a un dispositivo móvil.

En la imagen están señalizadas mediante números en color blanco sobre círculos de color rojo algunas características importantes del archivo. La siguiente tabla resume estas características:


| NÚMERO | DESCRIPCIÓN | ANTES | DESPUÉS |
|---|---|---|---|
| 1 | Peso del archivo | 65,1 MB | 3,10 MB |
| 2 | Tamaño del fotograma (Ancho x Alto) | 640 x 480 | 352 x 288 |
| 3 | Tasa de fotogramas | 30 fps | 15 fps |
| 4 | Formato original | | AVI |
| 5 | Dispositivo de salida | | Portátil |
| 6 | Formato de salida | | 3GP |

La operación de exportación puede llevar asociada una operación de conversión ya que debemos elegir el formato de destino de nuestro vídeo y el códec empleado para su compresión. Además, la mayoría de los vídeos están almacenados en un formato que contiene también una información de audio. Si es el caso, debemos especificar también el códec a emplear para comprimir el audio o si deseamos exportarlo sin audio.

### VLC

Otra herramienta muy interesante, es VLC, Este popular reproductor multimedia oculta dentro un montón de funcionalidades que nos pueden ser realmente útiles.

**Controles avanzados**

Si queremos sacar parte de un archivo de sonido o de música de forma rápida podemos usar los controles avanzados y grabar esas partes. Los controles se pueden mostrar en:

Menú > Ver > Controles avanzados

**Funcionalidades**

- Nos permite comenzar a grabar un fichero desde el punto actual hasta que se vuelva a pulsar
- Permite sacar un pantallazo del fotograma actual
- Crea un bucle de reproducción desde los dos puntos marcados con este mismo botón
- Si se pulsa, reproduce fotograma a fotograma

**Conversión**

Esta es una de las operaciones básicas sumamente útiles que ofrece VLC. Podemos convenir a múltiples formatos de audio y vídeo o incluso sacar el audio de un vídeo. Para iniciar el proceso de conversión, abrimos VLC y hacemos:

Menú > Medio > Convertir

Una vez ahí seleccionamos el contenido que queremos convertir y pulsamos el botón Convertir/Guardar.

## 2.5. INSERTAR VIDEO EN UNA WEB

Para aprender a insertar vídeo en páginas web se empezará por las posibilidades de HTML5. HTML5 ofrece soluciones mucho más ventajosas para insertar audio que su predecesor HTML4. Y para el caso del vídeo ocurre lo mismo. En HTML5 hay una etiqueta <video> que permite embeber archivos de vídeo de forma nativa sin necesidad de complementos (plugins) adicionales.

La etiqueta <video> es muy parecido a <audio>: dispone de los atributos **autoplay**, **loop** y **preload**, con la misma sintaxis y semántica que en <audio>. También se puede especificar la fuente de un archivo bien usando el atributo **src** o bien usando la etiqueta <source>. Así mismo, se puede utilizar los controles que ofrece el navegador de forma nativa utilizando el atributo controla o bien puedes ofrecer tus propios controles en JavaScript.

Sin embargo, a diferencia del audio, el vídeo necesita un tamaño para reproducirse, tal y como ocurre con las imágenes. Para ello se pueden usar los atributos **height** (alto) y **width** (ancho) con más sentido que en <audio>. Un ejemplo de uso de esta etiqueta con un archivo mp4 es el siguiente:

<video poster= "images/portada.png" src="videos/recetapollo .mp4" controls width="360" height="240"></video>

Además, de los atributos vistos, la etiqueta <video> también tiene un atributo **poster** en el que se indica una imagen que se usará como portada antes de que el vídeo empiece a reproducirse.

El ejemplo anterior usa un archivo mp4 lo que hace que su reproducción sea nativa solo en Safari y no en IExplorer, Firefox ni Google Chrome. Entonces, para hacer la reproducción adecuada para la mayor cantidad de navegadores posible, se puede utilizar la etiqueta <source> igual que se hizo

en <audio>. En el siguiente ejemplo se ha convertido el formato mp4 en ogv y webm con la utilidad Miro Converter.

```
<!DOCTYPE html>
<html>
  <head> </head>
  <body>
    <video controls="" width="360" height="240">
    <source src="videos/recetapollo.mp4" type="video/mp4">
    <!— Safari 5 —>
    <source src="videos/recetapollo.ogv" type="video/ogg;
                          codecs='theora, vorbis'">
          <!-- Firefox 5 y Google Chrome 14 -->
    <source src="videos/recetapollo.webm" type="video/webm">
    <!— Firefox 5 y Google Chrome 14 -->
  </video>
  </body>
</html>
```

Sin esta solución es solo válida para navegadores que soportan HTML5. Si se desea tener en cuenta a versión anteriores, por ejemplo, a lExplorer, se puede utilizar etiquetas <object> (o usar JavaScript). Esta solución es similar a la vista para audio.

```
<!DOCTYPE html>
<html>
  <head>
  </head>
  <body>
  <video controls="" width="360" height="240">
  <source src="videos/recetapollo.mp4" type="video/mp4">
  <!-- Safari 5 -->
  <source src="videos/recetapollo.ogv" type = "video/ogg; codecs = 'theora,
vorbis'"><!--Firefox 5 y Google Chrome 14 -->
<source src="videos/recetapollo.webm" type="video/webm">
<!-- Firefox 5 y Google Chrome 14 -->
<object type="application/x-shockwave-flash" data="player_flv_maxi.swf"
height="240" width="360">
<param name="movie" value="player_flv_maxi.swf">
<param name="FlashVars" value="flv=videos/recetapollo.flv">
<embed type="application/x-shockwave-flash" width="360" height="240"
src="player_flv_maxi.swf" flashvars="flv=videos/recetapollo.flv">
</embed>
  </object>
  </video>
```

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

Para que este código funcione es necesario que el archivo player_flv_maxi.swf esté disponible en la misma carpeta (local o en web) que la página html. Este archivo es el que carga <object> cuando ninguna de las otras opciones son soportadas. <object> tiene varios parámetros que definen el tamaño del elemento que quiere embeber, en este caso el reproductor Flash.

Es importante destacar, que no todos los vídeos que se insertan en páginas web deben estar en propiedad del creador. También es posible utilizar vídeos disponibles en cualquier repositorio de vídeo como, por ejemplo, Youtube o Vimeo. Estos repositorios ofrecen código para incrustar un vídeo determinado en una página web. Actualmente, el código proporcionado utiliza la etiqueta <iframe> en sustitución a <object> con el fin de dar soporte más optimizado a páginas web para dispositivos móviles tipos iPad.

Como ocurre en audio, también a la etiqueta <vídeo> se le pueden aplicar estilos con CSS.

## 2.6. EL API DE VIDEO

Para poder gestionar y manipular la reproducción de vídeo desde JavaScript disponemos de je API cuyos métodos y propiedades se resumen en la siguiente tabla:

| **M**étodos | Propiedades | Eventos |
|---|---|---|
| play() | currenteSrc | play |
| pause() | currentTime | pause |
| **load()** | startTime (read only) | progress |
| **canPlayTye()** | videoWidth | error |
| | videoHeight | timeupdate |
| | duration (read only) | ended |
| | ended (read only) | abort |
| | error | empty |
| | paused (read only) | emptied |
| | muted | waiting |
| | seeking | loadmetadata |
| | volume | |
| | height | |
| | width | |
| | seekable (read only) | |
| | played (read only) | |

A continuación, vamos a mostrar algunos ejemplos de uso de este API. La ventaja del vídeo HTML5 es que no deja de ser una etiqueta más, y por tanto, una vez seleccionado ese elemento podemos hacer lo que queramos.

## CONTROL DE REPRODUCCIÓN

En este sencillo ejemplo vemos cómo podemos controlar la reproducción desde JavaScript. Lo que haremos será añadir unos sencillos botones propios que al pulsarlos podremos controlar la reproducción del vídeo.

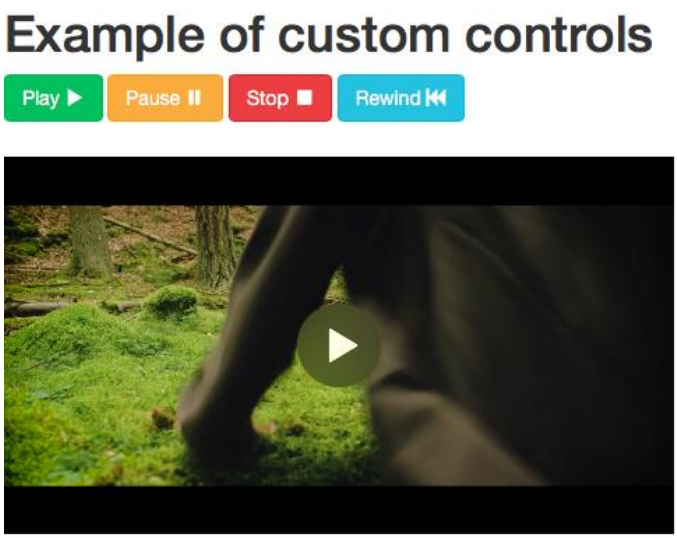
Nótese que no existe un método stop() para detener el vídeo, pero se simula llamando a rewind y pause. Para crear unos botones más atractivos y de paso aplicar lo visto hasta utiliza Bootstrap y los glyphicons.

**Código**

```
<!DOCTYPE HTML>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>Simple video control</title>
  <link rel="stylesheet" href="css/bootstrap.min.css" type="text/css">
</head>
<body>
    <header>
        <h1>Example of custom controls</h1>
        <button onclick="playVideo();" class="btn btn-success">
            Play <i class="glyphicon glyphicon-play"></i>
        </button>
        <button onclick="pauseVideo();" class="btn btn-warning">
            Pause <i class="glyphicon glyphicon-pause"></i>
        </button>
        <button onclick="stopVideo()" class="btn btn-danger">
            Stop <i class="glyphicon glyphicon-stop"></i>
        </button>
        <button onclick="rewindVideo();" class="btn btn-info">
            Rewind <i class="glyphicon glyphicon-fast-backward"></i>
        </button>
       </header>
<video width="480" height="320" controls="controls" id="myVideo">
   <source src="movie_clip.mp4" type="video/mp4" />
```

```
   <source src="movie_clip.ogg" type="video/ogg" />
   Your browser does not support the <video> element.
</video>

<script>
    theVideo = document.querySelector("#myVideo");

    function playVideo() {
       theVideo.play();
    }

    function pauseVideo() {
       theVideo.pause();
    }

    function stopVideo() {
       theVideo.currentTime = 0;
       theVideo.pause();
    }

    function rewindVideo() {
       theVideo.currentTime = 0;
    }
</script>
</body>
</html>
```

**Aspecto**




## EVENTOS

El vídeo puede generar varios eventos y podemos aplicar **listeners** para llevar a cabo distintas operaciones. En el siguiente ejemplo se controlan los eventos de inicio, de fin, etc. y de paso se nuestra cómo mostrar el progreso temporal del vídeo en un elemento de la página.

Para el progreso se utiliza un evento llamado **timeUpdate**. Este evento tiene un valor que da el paso del tiempo, pero ojo, no nos da el momento exacto del vídeo. Ese dato lo sacamos de la propiedad **currentTime**.

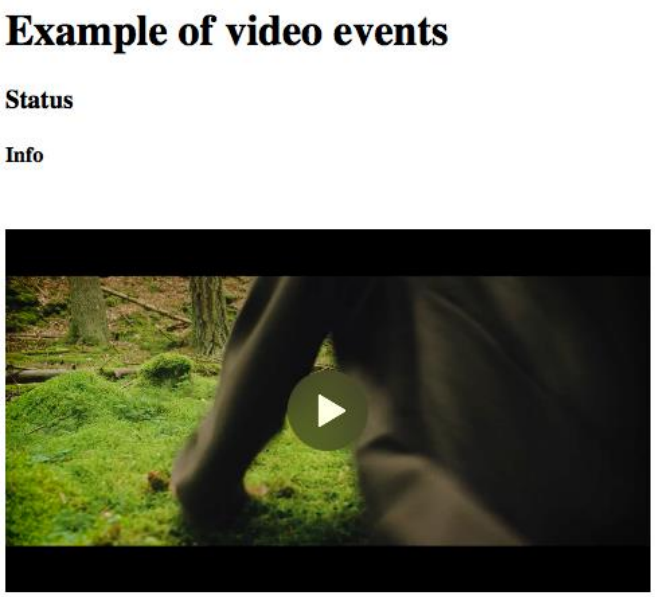
### Código

```
<!DOCTYPE HTML>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>Simple video events</title>
</head>
<body>
    <header>
        <h1>Example of video events</h1>
           <h3 id="status">Status</h3>
           <h4 id="info">Info</h4>
    </header>
<video width="480" height="320" controls="controls" id="myVideo">
```

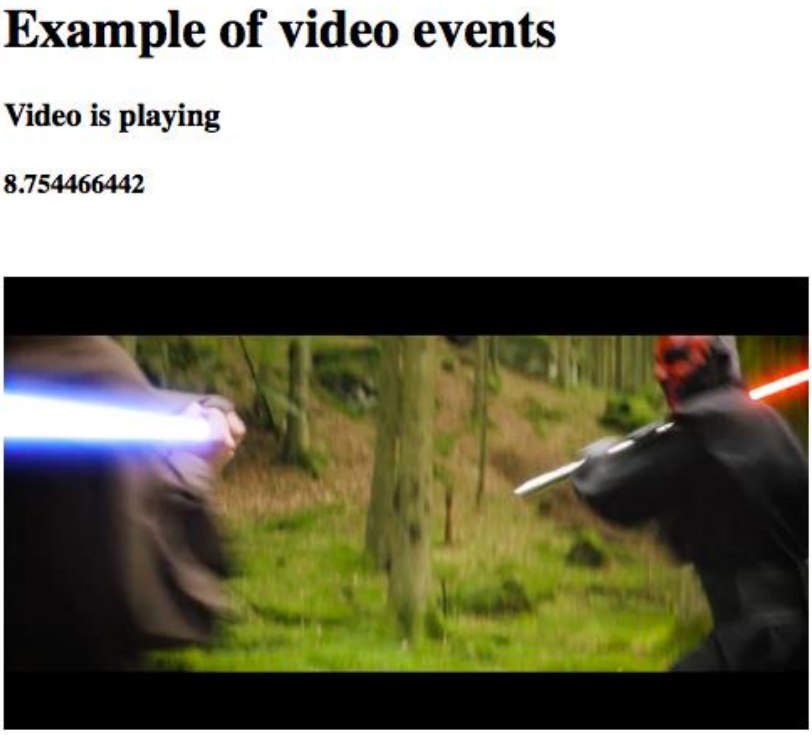
```
    <source src="movie_clip.mp4" type="video/mp4" />
    <source src="movie_clip.ogg" type="video/ogg" />
    Your browser does not support the <video> element.
</video>

<script type='text/javascript'>
  var theVideo = document.querySelector('#myVideo');

  // We add listeners
  theVideo.addEventListener('play', videoPlaying, false);
  theVideo.addEventListener('ended', videoEnded, false);
  theVideo.addEventListener('timeupdate', videoTimeUpdate, false);

  function videoPlaying (e) {
    var status = document.querySelector('#status');
    status.innerHTML = 'Video is playing';
  }

  function videoEnded (e) {
     var status = document.querySelector('#status');
     status.innerHTML = 'End of the video';
  }

  function videoTimeUpdate (e) {
      var info = document.querySelector('#info');
      info.innerHTML = theVideo.currentTime;
  }
</script>
</body>
</html>
```

**Aspecto**

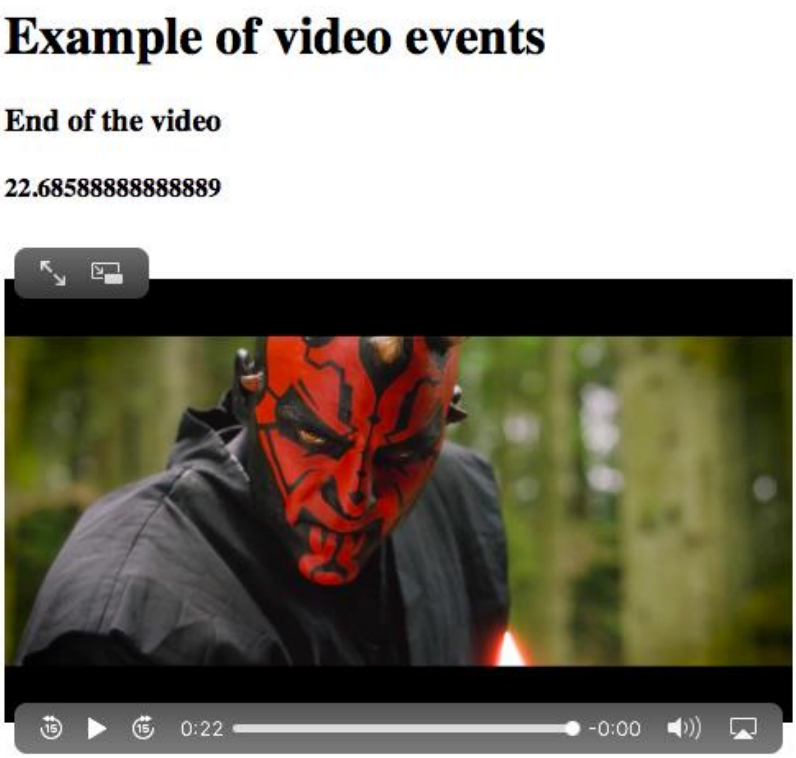
Al abrir la página:




Evento: playing




Evento: final del video (ended)




## LISTAS DE REPRODUCCIÓN

También podemos controlar qué vídeo se ve a través de JavaScript. Sabiendo esto, implantar una lista de reproducción resulta trivial. Basta con definir un **array** con varios ficheros de vídeo y pasaremos de uno a otro con los botones.

En la página mostramos la lista de ficheros de vídeo y unos controles básicos para pasar de uno a otro. En el código se muestran dos maneras de cargar el fichero de vídeo en el reproductor:

1. Modificando simplemente el atributo src del elemento <video>.
2. Añadiendo varias etiquetas source mediante un innerHTML.

En cualquiera de los dos casos no hay que olvidar llamar al método **load** para que el vídeo se cargue en el reproductor.

**Código**

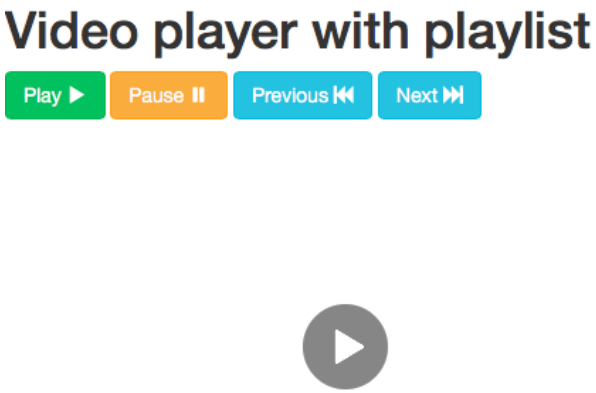
```
<!DOCTYPE HTML>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>Video player with playlist</title>
```

```
  <link rel="stylesheet" href="css/bootstrap.min.css" type="text/css">
</head>
<body>
    <header>
        <h1>Video player with playlist</h1>
        <button onclick="playVideo();" class="btn btn-success">
            Play <i class="glyphicon glyphicon-play"></i>
        </button>
        <button onclick="pauseVideo();" class="btn btn-warning">
            Pause <i class="glyphicon glyphicon-pause"></i>
        </button>
        <button onclick="previous()" class="btn btn-info">
            Previous <i class="glyphicon glyphicon-fast-backward"></i>
        </button>
        <button onclick="next ();" class="btn btn-info">
            Next <i class="glyphicon glyphicon-fast-forward"></i>
        </button>
        <div id="info"></div>
       </header>

    <video width="480" height="320" controls="controls" id="myVideo">
    Your browser does not support the <video> element.
    </video>

<script type='text/javascript'>
    theVideo = document.querySelector("#myVideo");
    info = document.querySelector("#info");
    var videos = ['movie_clip', 'movie_clip2', 'movie_clip3'];
    var current = 0;

    function playVideo() {
        // The simplest way:
        //theVideo.src = videos[current] + '.mp4';
```

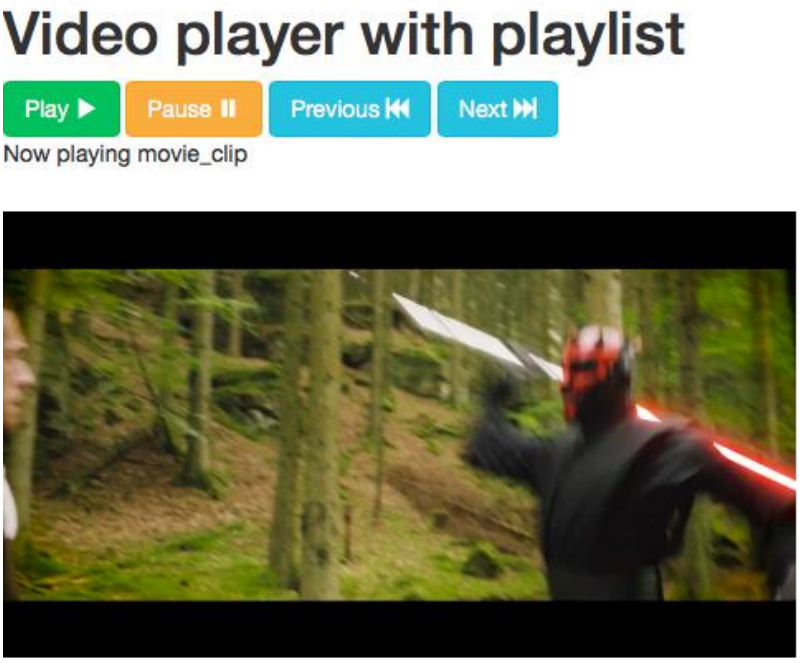
```
        // adding suppor for various formats
       theVideo.innerHTML = "";
       theVideo.innerHTML      =      "<source      src='"+videos[current]+".mp4'
type='video/mp4'>";
       theVideo.innerHTML      +=      "<source      src='"+videos[current]+".ogg'
type='video/ogg'>";
       theVideo.load();
       theVideo.play();
       info.innerHTML = 'Now playing ' + videos[current];
    }

    function pauseVideo() {
       theVideo.pause();
       info.innerHTML = videos[current] + ' paused';
    }

    function previous () {
        if (current > 0) {
            current--;
        }
        playVideo();
    }

    function next () {
        if (current < videos.length-1) {
            current++;
        }
        playVideo();
    }
</script>
</body>
</html>
```

**Aspecto**

Al abrir la página:




Al reproducir un video:




## 2.7. REPOSITORIOS DE VÍDEO

Actualmente Youtube es el líder en portales con servicio de vídeo en línea. Sin embargo, cada vez proliferan más los sitios web de este tipo donde es posible subir y visualizar contenidos de vídeo. En algunos de ellos también se pueden descargar vídeos al disco duro local para visualizarlo con los alumnos en aulas sin conexión a internet y también asegurando la actividad frente a las limitaciones de una conexión modesta.

En algunos casos se puede aplicar el plugin **Video DownloadHelper** de Mozilla Firefox y en otros casos en el mismo sitio se ofrece como alternativa la descarga directa del archivo de video.

A continuación, se citan algunos de los servicios de vídeos más conocidos donde se pueden descargar vídeos FLV y MP4 utilizando Firefox + VideoDownload Helper:

- **YouTube**: http://www.youtube.com
- **Mediateca de EducaMadrid**: http://mediateca.educa.madrid.org/
- **TeacherTube**: http://www.teachertube.com/
- **Metacafe**: http://www.metacafe.com/
- **MySpace Videos**: http://es.myspace.com/video
- **Dailymotion**: http://www.dailymotion.com/
- **Vimeo**: http://www.vimeo.com
- **Current TV**: http://current.com/
- **National Geographic**: http://video.nationalgeographic.com/video/

## ANEXO II. ELEMENTOS VIDEO Y AUDIO EN HTML5.

https://youtu.be/CKtI8y7kqyE