

Tema 5. Vídeo y audio.

1) Introducción. Etiquetas Audio y Vídeo	1
2) Formatos y compatibilidades.....	2
3) Etiqueta Vídeo.....	3
3) Etiqueta Source.....	4
4) Etiqueta Track.....	5
5) Etiqueta Audio.....	7
6) Control del Error en los elementos de vídeo y audio.	10
7) Creando nuestro propio reproductor de vídeo y audio.	12
Ejemplo.....	12
Ejemplo.....	15
Ejercicio 1:	16
Crear un reproductor personalizado para VÍDEO y AUDIO	16
Pasos a seguir	16

1) Introducción. Etiquetas Audio y Vídeo

Cada vez más el uso de elementos multimedia es muy importante.

Antes se utilizaba Flash para colocar animaciones, para ello se necesitaba tener un pequeño programa en nuestro equipo. Y el problema viene cuando no en todos los dispositivos se puede utilizar esta tecnología.

Pues bien, el objetivo de HTML5, es conseguir que colocar un elemento multimedia de vídeo o de audio en una página Web, sea una tarea de lo más sencilla, y que además, funcione perfectamente en todos los dispositivos. En estos momentos, estamos en el buen camino para conseguirlo, pero existen todavía muchos conflictos e intereses por parte de los más potentes fabricantes como veremos más adelante.

Con este objetivo por tanto, aparecen en HTML5 una serie de etiquetas que vamos a ir viendo en esta unidad y que son las siguientes:

<i>Etiqueta HTML</i>	<i>Definición</i>
<vídeo>	Etiqueta que define un vídeo o una película.
<audio>	Etiqueta que define el contenido de sonido.

<code><track></code>	Etiqueta que define las pistas de texto para los elementos multimedia.
<code><source></code>	Etiqueta que define los recursos de medios múltiples para <code><video></code> y <code><audio></code> .

2) Formatos y compatibilidades.

El objetivo, es que los propios navegadores contengan los codecs necesarios para reproducir todos los elementos que coloquemos mediante estas nuevas etiquetas.

Pero esto no es del todo cierto por el momento. Los intereses por parte de los principales desarrolladores, han desencadenado en un conflicto para decidir que codecs y que formatos utilizar para estos elementos. En un principio se pensó en un formato de compresión libre, pero dados los intereses de algunas partes esto no llegó a concretarse. Por el momento, los tres formatos mejor posicionados son los siguientes:

OGG THEORA, de formato libre abierto y gratuito. Cuando hablamos de OGG THEORA como un codec, no es del todo cierto. Digamos que THEORA es un codec de vídeo, y que OGG es la capa de transporte que está por encima. Por lo tanto, en el proyecto OGG tenemos el codec THEORA de vídeo, y el codec VORBIS de audio. Se considera de menor calidad que H.264 y VP8, pero ha recibido el apoyo de Firefox desde el principio.

H.264, un codec propietario. Apoyado desde el principio por los navegadores Chrome y Safari, y más tarde por IE9. Es el codec usado por plataformas como YouTube y Vimeo para la visualización de vídeos embebidos en HTML5 que ofrece un excelente nivel de calidad.

VP8, un codec propietario que compró Google y que liberó en código abierto. Presentado por On2 como un codec de alta calidad y un buen nivel de compresión. On2 Technologies fue comprada por Google, mejoró sus codecs y los liberó con el objetivo de tener unos codecs competentes en código abierto.

Teniendo en cuenta todo esto, por lógica, el codec que debería de triunfar es VP8, de muy buena calidad y libre. Pero como todos sabemos, los intereses económicos están por encima de todo, y por lo tanto, en estos momentos en mejor posicionado es H.264, el único que es propietario.

En cuanto a la compatibilidad de los navegadores con los formatos de vídeo que hemos explicado:

En cuanto a la compatibilidad de los navegadores con los formatos de audio que hemos explicado:

Es importante tener en cuenta, la clara tendencia de Safari(Apple) y IE9(Microsoft) hacia los codecs de pago, la de Firefox y Opera hacia los libres y Chrome(Google) que lo soporta todo.

- Algunos conversores disponibles on-line son: Media Converter, Free on line entre otros.

3) Etiqueta Vídeo.

El elemento que nos permite reproducir archivos de vídeo de una forma sencilla en HTML5, es la etiqueta de <VIDEO>, y soporta los siguientes atributos:

Atributos globales.

src: Atributo que especifica la dirección URL del archivo de vídeo a reproducir.

width: Atributo que establece el ancho del reproductor de vídeo.

height: Atributo que establece la altura del reproductor de vídeo.

autoplay: Atributo que especifica que el vídeo se reproducirá tan pronto como esté listo de manera automática.

controls: Atributo que especifica que los controles de vídeo deben de ser mostrados.

loop: Atributo que especifica que el vídeo se iniciará de nuevo, cada vez que termine la reproducción.

preload: Atributo que especifica si el autor piensa que el vídeo debe ser cargado cuando se carga la página y de qué manera.

poster: Atributo que especifica una imagen que se muestra mientras el vídeo se descarga, o hasta que el usuario pulsa el botón de reproducción.

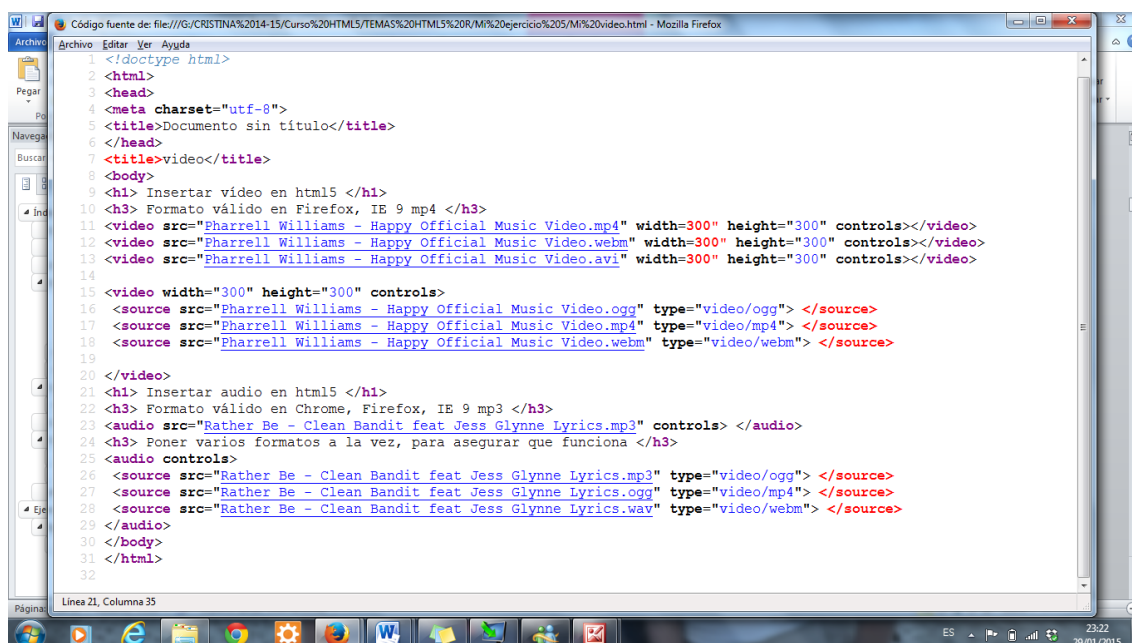
muted: Atributo que especifica que la salida de audio del vídeo debe ser silenciada.

crossorigin: Atributo que especifica si un archivo de vídeo está disponible en un dominio diferente.

mediagroup: Atributo que se utiliza para unir varios archivos multimedia para la reproducción sincronizada.

Nota: Funciona en todos los navegadores.

Ejemplo:



```
1 <!doctype html>
2 <html>
3 <head>
4 <meta charset="utf-8">
5 <title>Documento sin título</title>
6 </head>
7 <title>video</title>
8 <body>
9 <h1> Insertar video en html5 </h1>
10 <h3> Formato válido en Firefox, IE 9 mp4 </h3>
11 <video src="Pharrell Williams - Happy Official Music Video.mp4" width=300 height=300 controls></video>
12 <video src="Pharrell Williams - Happy Official Music Video.webm" width=300 height=300 controls></video>
13 <video src="Pharrell Williams - Happy Official Music Video.avi" width=300 height=300 controls></video>
14
15 <video width=300 height=300 controls>
16 <source src="Pharrell Williams - Happy Official Music Video.ogg" type="video/ogg"> </source>
17 <source src="Pharrell Williams - Happy Official Music Video.mp4" type="video/mp4"> </source>
18 <source src="Pharrell Williams - Happy Official Music Video.webm" type="video/webm"> </source>
19
20 </video>
21 <h1> Insertar audio en html5 </h1>
22 <h3> Formato válido en Chrome, Firefox, IE 9 mp3 </h3>
23 <audio src="Rather Be - Clean Bandit feat Jess Glynne Lyrics.mp3" controls> </audio>
24 <h3> Poner varios formatos a la vez, para asegurar que funciona </h3>
25 <audio controls>
26 <source src="Rather Be - Clean Bandit feat Jess Glynne Lyrics.mp3" type="video/ogg"> </source>
27 <source src="Rather Be - Clean Bandit feat Jess Glynne Lyrics.ogg" type="video/mp4"> </source>
28 <source src="Rather Be - Clean Bandit feat Jess Glynne Lyrics.wav" type="video/webm"> </source>
29 </audio>
30 </body>
31 </html>
32
```

Como hemos puesto un video mp4, solo funcionará en los navegadores que soporten los codecs H.264, que en este caso son IE9, SAFARI y CHROME, en los demás, detectará que hay un elemento de vídeo, pero no lo reproducirá.

Como hemos puesto un video webm, solo funcionará en los navegadores que soporten los codecs VP8, que en este caso son FIREFOX, OPERA y CHROME, en los demás, detectará que hay un elemento de vídeo, pero no lo reproducirá.

Por tanto, dependiendo del formato, se verá en un navegador o en otro, pero cuando funcione cada navegador mostrará su propio reproductor personalizado. Esto será lo que veremos en pantalla cuando el navegador sea capaz de reproducirlo:

Vemos por lo tanto, como en todos los casos el reproductor es diferente ya que depende del navegador que estemos utilizando. Pero en todos los casos los controles están presentes, ya que se lo hemos indicado mediante el atributo controls de la etiqueta <VIDEO>.

Nota: La etiqueta <video> no es suficiente para reproducir el vídeo en todos los navegadores. De momento no hay un solo formato que sea soportado por todos los navegadores.

3) Etiqueta Source.

Como hemos visto, con la etiqueta <VIDEO> solo podemos colocar un archivo de vídeo de un formato en concreto en nuestra página. Sin embargo, disponemos de esta etiqueta <SOURCE> que nos permite utilizar el mismo archivo con diferentes formatos para que cada navegador elija cuál es el formato que puede reproducir, y así conseguir que se pueda ver el archivo que queremos colocar en la página Web en cualquiera de los navegadores o dispositivos. Esta etiqueta soporta los siguientes atributos:

Atributos globales.

src: Atributo que especifica la dirección URL del medio a reproducir.

type: Atributo que especifica el tipo MIME del medio a reproducir.

media: Atributo que especifica el tipo de medio de recursos a reproducir.

Nota: Funciona en todos los navegadores.

Ejemplo

Utilizando el siguiente código:

```
<video width="300" height="300" controls>
  <source src="Pharrell Williams - Happy Official Music Video.ogg" type="video/ogg"> </source>
  <source src="Pharrell Williams - Happy Official Music Video.mp4" type="video/mp4"> </source>
  <source src="Pharrell Williams - Happy Official Music Video.webm" type="video/webm"> </source>
</video>
```

Vemos como en este caso hemos colocado el elemento de vídeo con las mismas características que en ejemplo anterior, pero en este caso no hemos definido el atributo src dentro de la etiqueta <VIDEO>, sino que dicho atributo se lo hemos definido dentro de cada etiqueta <SOURCE>. Vemos que tenemos tres elementos de este tipo, y en cada uno de ellos el fichero de vídeo es el mismo pero con diferente formato. De esta manera, el navegador recorrerá las fuentes hasta encontrar una que sea capaz de reproducir.

En este caso hemos colocado los tres principales formatos, y con ello nos aseguramos que en todas las versiones actualizadas de los navegadores con los que venimos trabajando funcione sin problema alguno.

4) Etiqueta Track.

Aparte de poder colocar un archivo de vídeo en varios formatos utilizando las etiquetas <VIDEO> y <SOURCE>, en HTML5 aparece esta potente etiqueta <TRACK> que permite definir pistas de texto para los elementos de vídeo, como pueden ser subtítulos, capítulos, o descriptores para lectores de pantalla. Está preparada para soportar un amplio abanico de posibilidades en cuanto a formatos y tipos de ficheros se refiere, pero en el ejemplo vamos a ver el formato *WEBVTT*, soportado por esta etiqueta y preparado para ello.

Para ver en funcionamiento esta etiqueta, no podemos ejecutar la página de manera local, sino que debemos hacerlo desde un servidor Web. En cuanto a disponibilidad, de momento solo está en funcionamiento en las últimas versiones de Chrome y en las versiones beta de IE10.

Esta etiqueta soporta los siguientes atributos:

Atributos globales.

src: Atributo que especifica la dirección URL del archivo de la pista.

default: Atributo que especifica que pista se activará por defecto si las preferencias del usuario no indican lo contrario.

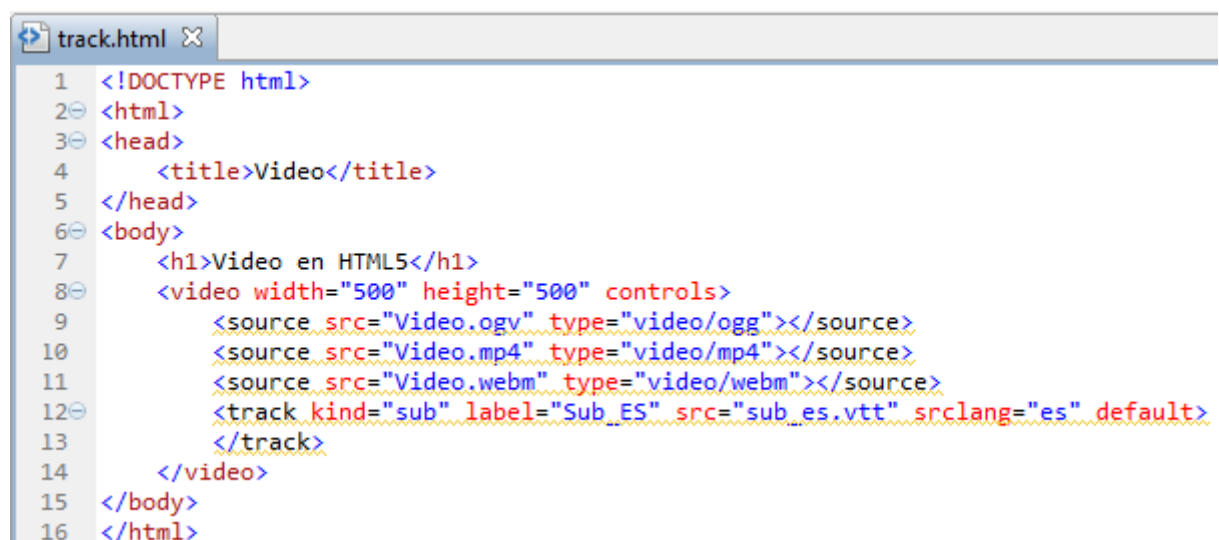
kind: Atributo que especifica el tipo de pista de texto.

label: Atributo que especifica el título de la pista de texto.

srclang: Atributo que especifica el idioma de los datos de la pista de texto.

Ejemplo

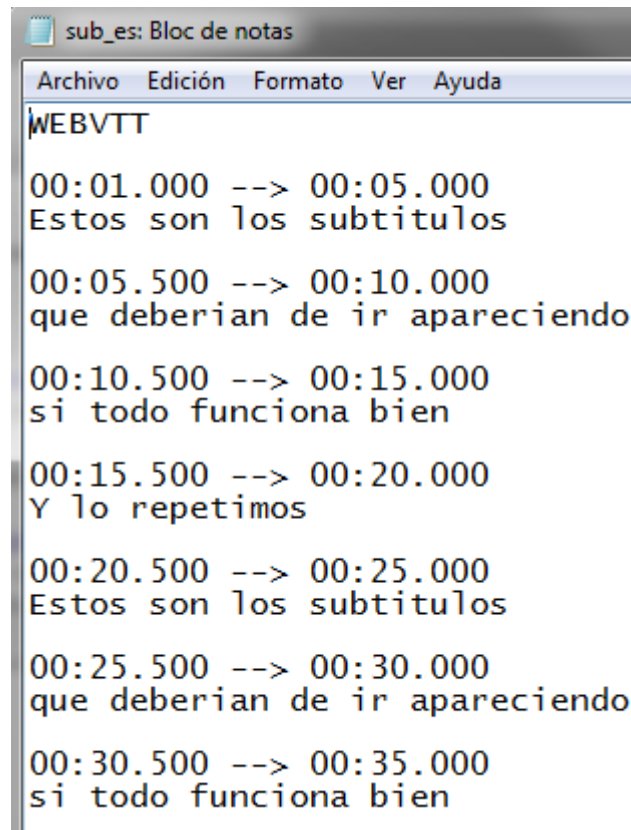
Utilizando el siguiente código:



```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <title>Video</title>
5 </head>
6 <body>
7   <h1>Video en HTML5</h1>
8   <video width="500" height="500" controls>
9     <source src="Video.ogv" type="video/ogg"></source>
10    <source src="Video.mp4" type="video/mp4"></source>
11    <source src="Video.webm" type="video/webm"></source>
12    <track kind="sub" label="Sub_ES" src="sub_es.vtt" srclang="es" default>
13  </track>
14 </video>
15 </body>
16 </html>
```

Vemos como tenemos un elemento de vídeo y mediante las etiquetas `<SOURCE>`, colocamos los diferentes formatos para que todos los navegadores puedan reproducirlo. Hemos incluido en este caso un elemento `<TRACK>` al final, pero dentro del elemento `<VIDEO>`. Como hemos dicho, este elemento define las pistas de texto que acompañaran al vídeo. En este caso, es el fichero de formato `WEBVTT` y con nombre `sub_es.vtt` el que hemos agregado mediante esta etiqueta. Mediante los atributos, le hemos indicado que es una pista de subtítulos, le hemos dado una etiqueta, le hemos indicado el idioma y le hemos dicho que sea la pista a reproducir por defecto.

Este es el formato que tiene el fichero `sub_es.vtt`.



```
sub_es: Bloc de notas
Archivo Edición Formato Ver Ayuda
WEBVTT

00:01.000 --> 00:05.000
Estos son los subtítulos

00:05.500 --> 00:10.000
que deberian de ir apareciendo

00:10.500 --> 00:15.000
si todo funciona bien

00:15.500 --> 00:20.000
Y lo repetimos

00:20.500 --> 00:25.000
Estos son los subtítulos

00:25.500 --> 00:30.000
que deberian de ir apareciendo

00:30.500 --> 00:35.000
si todo funciona bien
```

Lo primero es indicar el tipo de fichero que es, y luego ya agregamos el tiempo durante el cuál debe de estar presente el subtítulo y justo debajo el propio subtítulo. Y de esta forma vamos añadiendo tantos textos como queramos y los vamos distribuyendo a lo largo del tiempo que dura el vídeo.

5) Etiqueta Audio.

El elemento que nos permite reproducir archivos de audio de una forma sencilla en HTML5, es la etiqueta de `<AUDIO>`, y soporta los siguientes atributos:

Atributos globales.

src: Atributo que especifica la dirección URL del archivo de audio a reproducir.

autoplay: Atributo que especifica que el audio se reproducirá tan pronto como esté listo de manera automática.

controls: Atributo que especifica que los controles de audio deben de ser mostrados.

loop: Atributo que especifica que el audio se iniciará de nuevo, cada vez que termine la reproducción.

preload: Atributo que especifica si el autor piensa que el audio debe ser cargado cuando se carga la página y de qué manera.

muted: Atributo que especifica que la salida de audio debe ser silenciada.

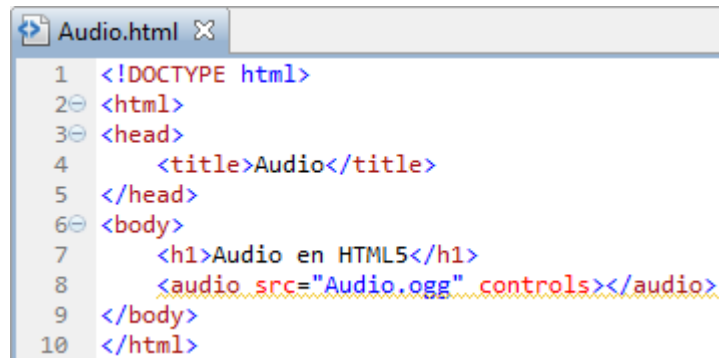
crossorigin: Atributo que especifica si un archivo de audio está disponible en un dominio diferente.

mediagroup: Atributo que se utiliza para unir varios archivos multimedia para la reproducción sincronizada.

Nota: Funciona en todos los navegadores

Ejemplo

Utilizando el siguiente código:



```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <title>Audio</title>
5 </head>
6 <body>
7   <h1>Audio en HTML5</h1>
8   <audio src="Audio.ogg" controls></audio>
9 </body>
10 </html>
```

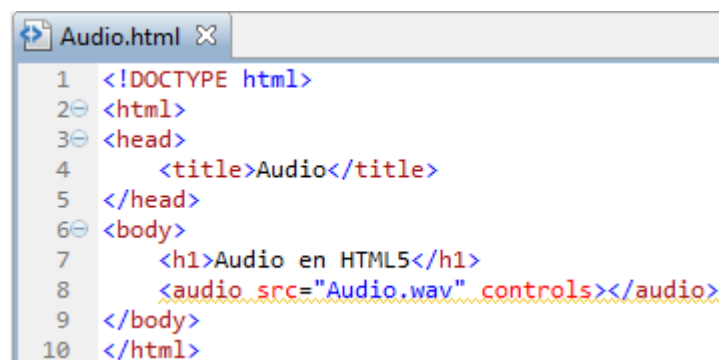
Vemos que hemos colocado un archivo de audio llamado *Audio.ogg*. Además, mediante el atributo *controls*, le indicamos que los controles para el audio tienen que estar activos. Como hemos indicado anteriormente, esto solo funcionará en los navegadores que soporten los codecs *OGG THEORA*, que en este caso son FIREFOX, OPERA y CHROME, en los demás, detectará que hay un elemento de audio, pero no lo reproducirá.

Si utilizáramos el siguiente código:

```
<h1> Insertar audio en html5 </h1>
<h3> Formato válido en Chrome, Firefox, IE 9 mp3 </h3>
<audio src="Rather Be - Clean Bandit feat Jess Glynne Lyrics.mp3" controls> </audio>
```

Solo funcionará en los navegadores que soporten los codecs *H.264*, que en este caso son IE9, SAFARI y CHROME, en los demás, detectará que hay un elemento de audio, pero no lo reproducirá.

Y si utilizáramos el siguiente código:



```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <title>Audio</title>
5 </head>
6 <body>
7   <h1>Audio en HTML5</h1>
8   <audio src="Audio.wav" controls></audio>
9 </body>
10 </html>
```

Solo funcionará en los navegadores que soporten los codecs *VP8*, que en este caso son FIREFOX, OPERA, SAFARI y CHROME, en los demás, detectará que hay un elemento de audio, pero no lo reproducirá.

Por tanto, dependiendo del formato, se verá en un navegador o en otro, pero cuando funcione cada navegador mostrará su propio reproductor personalizado. Esto será lo que veremos en pantalla cuando el navegador sea capaz de reproducirlo:

Vemos por lo tanto, como en todos los casos el reproductor es diferente ya que depende del navegador que estemos utilizando. Pero en todos los casos los controles están presentes, ya que se lo hemos indicado mediante el atributo controls de la etiqueta <AUDIO>.

Nota:

Solo con esta etiqueta <AUDIO>, no podremos hacer funcionar el elemento de audio en todos los navegadores.

De la misma manera que junto a la etiqueta de <VIDEO> podemos utilizar la etiqueta <SOURCE> para poder ofrecer el archivo en formatos diferentes y que el navegador pueda elegir el formato que él es capaz de reproducir, con la etiqueta de <AUDIO> pasa lo mismo, y podemos utilizar esa etiqueta <SOURCE> de la misma manera como vamos a ver en el siguiente ejemplo.

Ejemplo

Utilizando el siguiente código:

```
<h3> Poner varios formatos a la vez, para asegurar que funciona </h3>
<audio controls>
  <source src="Rather Be - Clean Bandit feat Jess Glynne Lyrics.mp3" type="video/ogg"> </source>
  <source src="Rather Be - Clean Bandit feat Jess Glynne Lyrics.ogg" type="video/mp4"> </source>
  <source src="Rather Be - Clean Bandit feat Jess Glynne Lyrics.wav" type="video/webm"> </source>
</audio>
```

Vemos como en este caso hemos colocado el elemento de audio con las mismas características que en el ejemplo anterior, pero en este caso no hemos definido el atributo src dentro de la etiqueta <AUDIO>, sino que dicho atributo se lo hemos definido dentro de cada etiqueta <SOURCE>. Vemos que tenemos tres elementos de este tipo, y en cada uno de ellos el fichero de audio es el mismo pero con diferente formato. De esta manera, el navegador recorrerá las fuentes hasta encontrar una que sea capaz de reproducir.

En este caso hemos colocado los tres principales formatos, y con ello nos aseguramos que en todas las versiones actualizadas de los navegadores con los que venimos trabajando funcione sin problema alguno. Lo que visualizará cada navegador, será lo que hemos visto en el ejemplo anterior, cada uno su reproductor personalizado.

Los **atributos globales**, son los siguientes:

accesskey: Para crear un atajo de teclado y acceder al elemento con él.

class: Clase a la que pertenece el comando.

contenteditable: Para hacer que el contenido sea editable desde el propio navegador.

contextmenu: Contiene el nombre del menú que será menú contextual del elemento.

dir: Especifica la direccionalidad del elemento de texto.

draggable: Para permitir que el elemento sea arrastrable.

dropzone: Para convertir el elemento en zona donde se puede arrastrar otro elemento.

hidden: Para convertir el elemento en oculto.

id: Definir el identificador para el elemento.

lang: Especifica el idioma principal del contenido del elemento.

spellcheck: Para permitir la corrección automática, se usa cuando el elemento es editable o cuando se puede escribir en él.

style: Atributo para definir el estilo del elemento.

tabindex: Para controlar cuándo se focaliza el elemento. El nombre viene de la tecla tabulador. Muy común en los formularios.

title: Define el título del elemento.

translate: Para definir si se traduce o no el elemento cuando se traduce la página desde el navegador.

6) Control del Error en los elementos de vídeo y audio.

Los elementos multimedia en general, y en este caso los elementos de audio y vídeo, pueden dar problemas en la ejecución o reproducción ya que como hemos visto hay que tener en cuenta varias cosas para que todo funcione de manera correcta. Pues bien, vamos a crear ahora una función que nos indicará en caso de fallo del elemento cuál ha sido el motivo del mismo. Siempre es más correcto hacer saltar una alerta indicando por qué no funciona, que dejar el elemento sin funcionar y sin ofrecer ningún tipo de detalle sobre el por qué. Además, conociendo el error siempre podemos tener más fácil la solución.

Vamos a crear por tanto una función que captará el error, y en función de cuál sea el motivo del mismo, mostrará una alerta u otra. Dicha función se ejecutará en el momento en que se detecte un error en los elementos de audio o vídeo, y es el evento **onerror** el que se activa cuando se detecta algún tipo de problema. Por tanto, en los elementos de audio o de vídeo, tendremos que indicar, que cuando suceda dicho evento se ejecute la función que vamos a crear y que nos indicará qué tipo de error ha sucedido.

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>Controlar el error</title>
<script>
function fail(e)
{
    switch (e.target.error.code)
    { case e.target.error.MEDIA_ERR_ABORTED:
      alert('Has abortado la reproducción');
      break;
      case e.target.error.MEDIA_ERR_NETWORK:
```

```

        alert('Error de red');
        break;
    case e.target.error.MEDIA_ERR_DECODE:
        alert('Problema de corrupción');
        break;
    case e.target.error.MEDIA_ERR_SRC_NOT_SUPPORTED:
        alert('Formato no soportado o archivo no
encontrado');
        break;
    default:
        alert('Ocurrió un problema desconocido');
        break;
    }
}
</script>
</head>

<body>
<h1> Insertar vídeo en html5 </h1>
<h3> Formato válido en Firefox, IE 9 mp4 </h3>
<video src="Pharrell Williams - Happy Official Music Video.mp4"
width=300" height="300" controls onerror="fail(event)">
</video>
</body>
</html>

```

Vemos como tanto en un elemento audio como en un elemento vídeo, definimos que la función que se debe ejecutar cuando se active el evento *onerror* sea la función *fail()*; dicha función es idéntica en ambos casos, y está definida de la siguiente manera:

Por tanto, si utilizamos esta función unida al evento *onerror*, en el momento que en el elemento audio o vídeo se detecta un error, dicha función es llamada, y le pasamos como parámetro el objeto *event* que ha desencadenado el error. Por tanto, en la función, cogemos el código del error tal y como vemos en la imagen, y en función de cuál sea dicho código, el error ha sido uno u otro, y mostramos la alerta acorde con el error.

Por ejemplo, si el formato del archivo no es soportado por el navegador que lo ejecuta, *e.target.error.code* se corresponderá con el error *MEDIA_ERR_SRC_NOT_SUPPORTED* y por tanto mostrará la alerta "Formato no soportado o archivo no encontrado".

Nota:

Si estamos controlando el error, nos dirán por qué no se reproduce. En los navegadores más antiguos que no soportan estas etiquetas, tenemos la opción de poner un texto que aparecerá en los casos en los que el navegador no sea capaz de interpretar las etiquetas.

Aunque en el ejemplo aparezca con la etiqueta *<VIDEO>*, en el caso de un elemento de *<AUDIO>* funcionará de la misma manera.

7) Creando nuestro propio reproductor de vídeo y audio.

Hemos visto como cada navegador presenta su propio reproductor personalizado para las etiquetas de `<VIDEO>` y `<AUDIO>`. Pues bien, nosotros también vamos a ser capaces de configurar un reproductor, o mejor dicho, una barra de controles personalizada a nuestro gusto, para nuestros elementos multimedia. Si nos paramos a mirar el DOM (Document Object Model) que tiene HTML5 para estas etiquetas, veremos que las posibilidades que nos ofrecen para controlar un elemento de vídeo o de audio desde un lenguaje de script a nuestro gusto son bastante amplias.

En este caso, vamos a ver los métodos y propiedades más importantes que disponemos para crear un reproductor sencillo, pero totalmente personalizado.

Los métodos más importantes son los siguientes:

play(): Método que comienza la reproducción del elemento de vídeo o de audio.

pause(): Método que detiene la reproducción del elemento de vídeo o de audio.

load(): Método que recarga el elemento de vídeo o de audio.

Y las propiedades más importantes son las siguientes:

duration: Propiedad que nos devuelve la duración del archivo de vídeo o de audio en segundos.

currentTime: Propiedad que nos devuelve la posición de la reproducción del archivo de vídeo o de audio en segundos.

ended: Propiedad que nos indica si la reproducción ha terminado.

muted: Propiedad que nos indica si la reproducción está silenciada.

paused: Propiedad que nos indica si la reproducción ha sido pausada.

volume: Propiedad que nos indica el nivel del volumen del 0 al 1.

Estas son solo algunas de las opciones que nos ofrece el DOM para estas etiquetas, pero con ellas vamos a ser capaces de crear una barra de controles totalmente personalizada como vamos a ver a continuación.

Ejemplo

En este primer ejemplo, vamos a crear una barra de controles para un elemento de vídeo.

En un primer paso, vamos a **colocar el elemento de vídeo**, y vamos a ir **creando todos los botones** necesarios para nuestro reproductor.

Este sería el código utilizado:

```
Código fuente de file:///G:/CRISTINA%202014-15/Curso%20HTML5/TEMAS%20HTML5%20R/M/%20ejercicio%205/M/%20reproductor%20de%20video%20y%20de%20audio.html - Mozilla Firefox
Archivo  Editar  Ver  Ayuda

1 </doctype html>
2 <html>
3 <head>
4 <meta charset="utf-8">
5 <title>Documento sin título</title>
6 <script>
7     function play()
8     {
9         var video = document.getElementById("miVideo");
10        video.play();
11    }
12    function pause()
13    {
14        var video = document.getElementById("miVideo");
15        video.pause();
16    }
17    function stop()
18    {
19        var video = document.getElementById("miVideo");
20        video.load();
21    }
22    function volumen()
23    {
24        var video = document.getElementById("miVideo");
25        var miVolumen = document.getElementById('volumen').value;
26        video.volume=miVolumen;
27    }
28    function silenciar()
29    {
30        var video = document.getElementById("miVideo");
31        if (video.muted == true)
32        { video.muted=false;
33          document.getElementById("btn_mute").innerHTML="SILENCIO=OFF";
34        }
35        else
36        { video.muted= true;
37          document.getElementById("btn_mute").innerHTML="SILENCIO=ON";
38        }
39    }
40 </script>
```

```
33        document.getElementById("btn_mute").innerHTML="SILENCIO=OFF";
34    }
35    else
36    { video.muted= true;
37      document.getElementById("btn_mute").innerHTML="SILENCIO=ON";
38    }
39  }
40 </script>
41 </head>
42
43 <body>
44 <video width="300" height="300" id="miVideo">
45   <source src="Pharrell Williams - Happy Official Music Video.ogv" type="video/ogv"> </source>
46   <source src="Pharrell Williams - Happy Official Music Video.mp4" type="video/mp4"> </source>
47   <source src="Pharrell Williams - Happy Official Music Video.webm" type="video/webm"> </source>
48 </video>
49 <div>
50   <button id="btn_play" onclick="play()">PLAY</button>
51   <button id="btn_pause" onclick="pause()">PAUSE</button>
52   <button id="btn_stop" onclick="stop()">STOP</button>
53   <input type="range" min="0" max="1" step="0.1" id="volumen" onchange="volumen()">
54   <button id="btn_mute" onclick="silenciar()">SILENCIO=OFF</button>
55 </div>
56 </body>
57 </html>
```

Línea 43, Columna 7

Hemos creado 4 botones y un input de tipo range, no hemos puesto el atributo controls, ya que no queremos que aparezca el reproductor que cada navegador establece por defecto.

El primero, será el botón que inicie la reproducción, y por tanto ejecutará la función *play()* al ser pulsado. El segundo, será el botón que pause la reproducción, y por tanto ejecutará la función *pause()* al ser pulsado. El tercero, será el botón que reinicie la reproducción, y por tanto ejecutará la función *stop()* al ser pulsado. El siguiente elemento, será un elemento *<input>* de tipo *range*, que controlará el volumen de la reproducción, y por tanto, cada vez que cambie su valor ejecutará la función *volumen()*. Y el último, será el botón que silencie la reproducción, y por tanto ejecutará la función *silenciar()* al ser pulsado

Y este el resultado que veríamos en pantalla:



Vemos como en lugar del reproductor por defecto que establece cada navegador, tenemos nuestra propia barra de controles.

Ahora, vamos a ir **creando las funciones** que se ejecutarán cuando hagamos clic sobre los botones, y que serán las que ejecuten y controlen los métodos y propiedades del DOM para controlar nuestro elemento de vídeo.

Creamos la función **play()**:

```
function play()
{
    var video = document.getElementById("miVideo");
    video.play();
}
```

Esta función se ejecutará al pulsar el botón de *PLAY*, y como vemos lo que hará será coger el elemento de vídeo que hemos colocado en nuestra página y que hemos identificado como "miVideo", y ejecutar el método *play()* del DOM. Por lo tanto, iniciará la reproducción del vídeo. Hay que aclarar, que el nombre de la función no tiene por qué ser el mismo que el del método.

Creamos la función **pause()**:

```
function pause()
{
    var video = document.getElementById("miVideo");
    video.pause();
}
```

Esta función se ejecutará al pulsar el botón de *PAUSE*, y como vemos lo que hará será coger el elemento de vídeo que hemos colocado en nuestra página y que hemos identificado como "miVideo", y ejecutar el método *pause()* del DOM. Por lo tanto, pausará la reproducción del vídeo. Hay que aclarar, que el nombre de la función no tiene por qué ser el mismo que el del método.

Creamos la función **stop()**:

```
function stop()
{
    var video = document.getElementById("miVideo");
    video.load();
}
```

Esta función se ejecutará al pulsar el botón de *STOP*, y como vemos lo que hará será coger el elemento de vídeo que hemos colocado en nuestra página y que hemos identificado como "miVideo", y ejecutar el método *load()* del DOM. Por lo tanto, reiniciará la reproducción del vídeo.

Creamos la función **volumen()**:

```
function volumen()
{
    var video = document.getElementById("miVideo");
    var miVolumen =
document.getElementById('volumen').value;
    video.volume=miVolumen;
}
```

Esta función se ejecutará cuando cambie el elemento de entrada de tipo range, y como vemos lo que hará será tomar el elemento de vídeo que hemos colocado en nuestra página y que hemos identificado como "miVideo", tomar el valor del elemento de entrada de tipo range que hemos identificado como "volumen", y asignar dicho valor, a la propiedad *volume* del DOM. Por lo tanto, cambiará el volumen de la reproducción del vídeo según el valor del elemento range.

Creamos la función **silenciar()**:

```
function silenciar()
{
    var video = document.getElementById("miVideo");
    if (video.muted == true)
    {
        video.muted=false;

        document.getElementById("btn_mute").innerHTML="SILENCIO=OFF";
    }
    else
    {
        video.muted= true;

document.getElementById("btn_mute").innerHTML="SILENCIO=ON";
    }
}
```

Esta función se ejecutará al pulsar el botón de *SILENCIO*, y como vemos lo que hará será tomar el elemento de vídeo que hemos colocado en nuestra página y que hemos identificado como "miVideo", y en caso de que la propiedad muted del DOM esté activa, la desactivará y en el botón pondrá "SILENCIO=OFF". En el caso de que la propiedad *muted* del DOM esté desactivada, la activará y en el botón pondrá "SILENCIO=ON". Por lo tanto, silenciará o activará el audio de la reproducción del vídeo.

De esta manera, tendremos ya controladas las principales funciones sobre nuestro elemento de vídeo.

Ejemplo

Y ahora, vamos a crear una barra de controles idéntica, pero en este caso para un elemento de audio.

En un primer paso, vamos a **colocar el elemento de audio**, y vamos a ir **creando todos los botones** necesarios para nuestro reproductor.

Vemos que la estructura es exactamente la misma que en el ejemplo de vídeo, solo que ahora, hemos colocado un elemento de audio en lugar de uno de vídeo, pero todo lo demás es exactamente igual.

Y este es el resultado que veríamos en pantalla:



Vemos como ahora también, en lugar del reproductor por defecto que establece cada navegador, tenemos nuestra propia barra de controles.

Todos los pasos siguientes, es decir, todas las funciones que controlan el elemento mediante los métodos y propiedades del DOM, son iguales que en el caso del elemento de vídeo. Solo que, el elemento que en el ejemplo de vídeo era "miVideo", ahora será "miAudio", ya que dichas funciones las ejecutaremos sobre el elemento de audio en este caso.

Nota:

Cuando creamos un reproductor personalizado, no solo conseguimos que este todo a nuestro gusto, sino que también podemos lograr que el reproductor sea idéntico para todos los navegadores.

Ejercicio 1:

Crear un reproductor personalizado para VÍDEO y AUDIO

Vamos a crear un reproductor personalizado para los elementos de vídeo y de audio. Utilizando las opciones que nos ofrece el DOM de HTML5 para estas etiquetas, vamos a crear una barra de controles.

Pasos a seguir

1. Opcional: La estructura de los reproductores, se puede hacer mediante una tabla.
2. La primera parte, la sección de los botones, en ambos reproductores, será muy similar a la que hemos visto en el ejemplo de la unidad.
3. La barra de progreso de ambos reproductores, la crearemos con un elemento `<progress>`, nuevo en HTML5. Y se irá actualizando a medida que avance la reproducción.
4. En el espacio del tiempo, se irá actualizando el tiempo transcurrido de la reproducción, y en todo momento aparecerá el la duración total de la reproducción. (Utilizaremos para ellos las propiedades `duration` y `currentTime` del DOM) Puedes consultar www.w3schools.com.
5. El formato del reproductor se lo asignaremos utilizando CSS.