## Programando un reproductor de video

Si has probado los anteriores códigos en diferentes navegadores, seguramente habrá notado que los diseños gráficos de los controles del reproductor difieren de uno a otro.

Cada navegador tiene sus propios botones y barras de progreso, e incluso sus propias funciones. Esta situación puede ser aceptable en algunas circunstancias, pero en un ambiente profesional, donde cada detalle cuenta, resulta absolutamente necesario que un diseño consistente sea preservado a través de dispositivos y aplicaciones, y también disponer de un control absoluto sobre todo el proceso.

HTML5 proporciona nuevos eventos, propiedades y métodos para manipular video e integrarlo al documento. De ahora en más, podremos crear nuestro propio reproductor de video y ofrecer las funciones que queremos usando HTML, CSS y Javascript. El video es ahora parte integral del documento.

### El diseño

Todo reproductor de video necesita un panel de control con al menos algunas funciones básicas. En la nueva plantilla del Listado 5-4, un elemento **<nav>** fue agregado luego de **<video>**. Este elemento **<nav>** contiene dos elementos ** ** (**botones** y **barra**) para ofrecer un botón "Reproducir" y una barra de progreso.

```
<!DOCTYPE html>
<html lang="es">
<head>
<title>Reproductor de Video</title>
<link rel="stylesheet" href="reproductor.css">
<script src="reproductor.js"></script>
</head>
<body>
<section id="reproductor">
<video id="medio" width="720" height="400">
<source src="metroMoscu.mp4">
</video>
<nav>
<div id="botones">
<button type="button" id="reproducir">Reproducir</button>
</div>
<div id="barra">
<div id="progreso"></div>
</div>
<div style="clear: both"></div>
</nav>
</section>
</body>
</html>
```

***Listado 5-4.*** *Plantilla HTML para nuestro reproductor de video.*

Además del video, esta plantilla también incluye dos archivos para acceder a códigos externos. Uno de ellos es **player.css** para los siguientes estilos CSS:

```
body{
text-align: center;
}
header, section, footer, aside, nav, article, figure, figcaption,
hgroup{
display : block;
}
#reproductor{
width: 720px;
margin: 20px auto;
padding: 5px;
background: #999999;
border: 1px solid #666666;
```

```
-moz-border-radius: 5px;
-webkit-border-radius: 5px;
border-radius: 5px;
}
nav{
margin: 5px 0px;
}
#botones{
float: left;
width: 100px;
height: 20px;
}
#barra{
position: relative;
float: left;
width: 600px;
height: 16px;
padding: 2px;
border: 1px solid #CCCCCC;
background: #EEEEEE;
}
#progreso{
position: absolute;
width: 0px;
height: 16px;
background: rgba(0,0,150,.2);
}
```

***Listado 5-5.*** *Estilos CSS para el reproductor.*

**Hágalo usted mismo:** Copie la nueva plantilla del Listado 5-4 en el archivo HTML (**video.html**). Cree dos nuevos archivos vacíos para los estilos CSS y el código Javascript. Estos archivos deberían ser llamados **reproductor.css** y **reproductor.js** respectivamente. Copie el código del Listado 5-5 dentro del archivo correspondiente y luego haga lo mismo para cada código Javascript listado de ahora en adelante.

**El código**

Es momento de escribir el código Javascript para nuestro reproductor. Existen diferentes formas de programar un reproductor de video, pero en este capítulo vamos solo a explicar cómo aplicar los necesarios eventos, métodos y propiedades para procesamiento básico de video. Vamos a trabajar con unas pocas funciones simples que nos permitirán reproducir y pausar el video, mostrar una barra de progreso mientras el video es reproducido y ofrecer la opción de hacer clic sobre esta barra para adelantar o retroceder el video.

**Los eventos**, estos son los más relevantes:

**progress** Este evento es disparado periódicamente para informar acerca del progreso de la descarga del medio. La información estará disponible a través del atributo **buffered**, como veremos más adelante.
**canplaythrough** Este evento es disparado cuando el medio completo puede ser reproducido sin interrupción. El estado es establecido considerando la actual tasa de descarga y asumiendo que seguirá siendo la misma durante el resto del proceso. Existe otro evento más para este propósito, **canplay**, pero no considera toda la situación y es disparado tan pronto como algunas partes del medio se encuentran disponibles (luego de descargar los primeros cuadros de un video, por ejemplo).
**ended** Es disparado cuando el reproductor llega al final del medio.
**pause** Es disparado cuando el reproductor es pausado.
**play** Es disparado cuando el medio comienza a ser reproducido.
**error** Este evento es disparado cuando ocurre un error. Es relacionado con el elemento **<source>** correspondiente a la fuente del medio que produjo el error.

Para nuestro reproductor de ejemplo solo vamos a escuchar a los habituales eventos **click** y **load**.

```
function iniciar() {
maximo=600;
medio=document.getElementById('medio');
reproducir=document.getElementById('reproducir');
barra=document.getElementById('barra');
progreso=document.getElementById('progreso');
reproducir.addEventListener('click', presionar, false);
barra.addEventListener('click', mover, false);
}
```
***Listado 5-6.*** *Función inicial.*

El Listado 5-6 presenta la primera función de nuestro reproductor de video. La función fue llamada **iniciar** debido a que será la función que iniciará la ejecución de la aplicación tan pronto como el documento sea completamente cargado.

Debido a que esta es la primera función a ser ejecutada, necesitamos definir unas variables globales para configurar nuestro reproductor. Usando el selector **getElementById** creamos una referencia a cada uno de los elementos del reproductor para poder acceder a ellos en el resto del código más adelante. También declaramos la variable **maximo** para conocer siempre el máximo tamaño posible para la barra de progreso (600 pixeles).

Hay dos acciones a las que tenemos que prestar atención desde el código: cuando el usuario hace clic sobre el botón "Reproducir" y cuando hace clic sobre la barra de progreso para avanzar o retroceder el video. Dos escuchas para el evento **click** fueron agregadas con el propósito de controlar estas situaciones. Primero agregamos la escucha al elemento **reproducir** que ejecutará la función **presionar()** cada vez que el usuario haga clic sobre el botón "Reproducir". La otra escucha es para el elemento **barra**. En este caso, la función **mover()** será ejecutada cada vez que el usuario haga clic sobre la barra de progreso.

**Los métodos**

La función **presionar()** incorporada en el Listado 5-7 es la primera función que realmente realiza una tarea. Esta función ejecutará de acuerdo a la situación actual dos métodos específicos de esta API: **play()**y **pause()**:

```
function presionar(){
if(!medio.paused && !medio.ended) {
medio.pause();
reproducir.innerHTML='Reproducir';
window.clearInterval(bucle);
}else{
medio.play();
reproducir.innerHTML='Pausa';
bucle=setInterval(estado, 1000);
}
}
```
***Listado 5-7.*** *Esta función inicia y pausa la reproducción del video.*

Los métodos **play()** y **pause()** son parte de una lista de métodos incorporados por HTML5 para procesamiento de medios. Los siguientes son los más relevantes:

**play()** Este método comienza a reproducir el medio desde el inicio, a menos que el medio haya sido pausado previamente.

**pause()**Este método pausa la reproducción.

**load()**Este método carga el archivo del medio. Es útil en aplicaciones dinámicas para cargar el medio anticipadamente.

**canPlayType(formato)** Con este método podemos saber si el formato del archivo es soportado por el navegador o no.

**Las propiedades**

La función **presionar()** también usa unas pocas propiedades para recabar información sobre el medio. Las siguientes son las más relevantes:
**paused** Esta propiedad retorna **true** (verdadero) si la reproducción del medio está actualmente pausada o no a comenzado.
**ended** Esta propiedad retorna **true** (verdadero) si la reproducción del medio ha finalizado porque se llegó al final.
**duration** Esta propiedad retorna la duración del medio en segundos.
**currentTime** Esta es una propiedad que puede retornar o recibir un valor para informar sobre la posición en la cual el medio está siendo reproducido o especifica una nueva posición donde continuar reproduciendo.
**error** Esta propiedad retorna el valor del error ocurrido.
**buffered** Esta propiedad ofrece información sobre la parte del archivo que ya fue cargada en el buffer. Nos permite crear un indicador para mostrar el progreso de la descarga. La propiedad es usualmente leída cuando el evento **progress** es disparado. Debido a que los usuarios pueden forzar al navegador a cargar el medio desde diferentes posiciones en la línea de tiempo, la información retornada por **buffered** es un array conteniendo cada parte del medio que ya fue descargada, no solo la que comienza desde el principio. Los elementos del array son accesibles por medio de los atributos **end()** y **start()**. Por ejemplo, el código **buffered.end(0)** retornará la duración en segundos de la primera porción del medio encontrada en el buffer. Esta propiedad y sus atributos están bajo desarrollo en este momento.

**El código en operación**
Ahora que ya conocemos todos los elementos involucrados en el procesamiento de video, echemos un vistazo a cómo trabaja la función **presionar()**.

Esta función es ejecutada cuando el usuario presiona el botón "Reproducir" en nuestro reproductor. Este botón tendrá dos propósitos: mostrará el mensaje "Reproducir" para reproducir el video o "Pausa" para detenerlo, de acuerdo a las circunstancias. Por lo tanto, cuando el video fue pausado o no comenzó, presionar este botón comenzará o continuará la reproducción. Lo opuesto ocurrirá si el video está siendo reproducido, entonces presionar el botón pausará el video. Para lograr esto el código detecta la situación del medio comprobando el valor de las propiedades **paused** y **ended**. En la primera línea de la función tenemos un condicional **if** para este propósito. Si el valor de **medio.paused** y **medio.ended** es falso, significará que el video está siendo reproducido, entonces el método **pause()** es ejecutado para pausar el video y el texto del botón es cambiado a "Reproducir" usando **innerHTML**. Si lo opuesto ocurre, el video fue pausado previamente o terminó de ser reproducido, entonces la condición será falsa (**medio.paused** o **medio.ended** es verdadero) y el método **play()** es ejecutado para comenzar o restaurar la reproducción del video. En este caso también realizamos una importante acción que es configurar un intervalo usando **setInterval()** para ejecutar la función **estado()** una vez por segundo de ahora en más.

```
function estado(){
if(!medio.ended){
var total=parseInt(medio.currentTime*maximo/medio.duration);
progreso.style.width=total+'px';
}else{
progreso.style.width='0px';
reproducir.innerHTML='Reproducir';
window.clearInterval(bucle);
}
}
```
***Listado 5-8.*** *Esta función actualiza la barra de progreso una vez por segundo.*

La función **estado()** en el Listado 5-8 es ejecutada cada segundo mientras el video es reproducido. También utilizamos un condicional **if** en esta función para controlar el estado del video. Si la propiedad **ended** retorna

falso, calculamos qué tan larga la barra de progreso debe ser en pixeles y asignamos el valor al elemento ** ** que la representa.

En caso de que la propiedad sea verdadera (lo cual significa que la reproducción del video ha terminado), retornamos el valor de la barra de progreso a 0 pixeles, cambiamos el botón a "Reproducir", y cancelamos el intervalo usando **clearInterval**. En este caso la función **estado()** no será ejecutada nunca más.

Volvamos unos pasos para estudiar cómo calculamos el tamaño de la barra de progreso. Debido a que la función **estado()** será ejecutada cada segundo mientras el video se está reproduciendo, el valor del tiempo en el que el video se encuentra cambiará constantemente.

Este valor en segundos es obtenido de la propiedad **currentTime**. También contamos con el valor de la duración del video en la propiedad **duration**, y el máximo tamaño de la barra de progreso en la variable **maximo** que definimos al principio. Con estos tres valores podemos calcular cuántos pixeles de largo la barra debería ser para representar los segundos ya reproducidos. La fórmula **tiempo actual × maximo / duración total** transformará los segundos en pixeles para cambiar el tamaño del elemento ** ** que representa la barra de progreso.

La función para responder al evento **click** del elemento **reproducir** (el botón) ya fue creada. Ahora es tiempo de hacer lo mismo para responder a los clics hechos sobre la barra de progreso:

```
function mover(e){
if(!medio.paused && !medio.ended){
var ratonX=e.pageX-barra.offsetLeft;
var nuevoTiempo=ratonX*medio.duration/maximo;
medio.currentTime=nuevoTiempo;
progreso.style.width=ratonX+'px';
}
}
```

***Listado 5-9.*** *Comenzar a reproducir desde la posición seleccionada por el usuario.*

Una escucha para el evento **click** fue agregada al elemento **barra** para responder cada vez que el usuario quiera comenzar a reproducir el video desde una nueva posición.

La escucha usa la función **mover()** para responder al evento cuando es disparado. Puede ver esta función en el Listado 5-9. Comienza con un **if**, al igual que las anteriores funciones, pero esta vez el objetivo es controlar que la acción se realice sólo cuando el video está siendo reproducido. Si las propiedades **paused** y **ended** son falsas significa que el video está siendo reproducido y el código tiene que ser ejecutado.

Debemos hacer varias cosas para calcular el tiempo en el cual el video debería comenzar a ser reproducido. Necesitamos determinar cuál era la posición del ratón cuando el clic sobre la barra fue realizado, cuál es la distancia en pixeles desde esa posición hasta el comienzo de la barra de progreso y cuantos segundos esa distancia representa en la línea de tiempo.

Los procesos para agregar una escucha (o registrar un evento), tales como **addEventListener()**, siempre envían un valor que hacer referencia al evento. Esta referencia es enviada como un atributo a la función que responde al evento.

Tradicionalmente la variable **e** es usada para almacenar este valor. En la función del Listado 5-9 usamos esta variable y la propiedad **pageX** para capturar la posición exacta del puntero del ratón al momento en el que el clic fue realizado. El valor retornado por **pageX** es relativo a la página, no a la barra de progreso o la ventana. Para saber cuántos pixeles hay desde el comienzo de la barra de progreso y la posición del puntero, tenemos que substraer el espacio entre el lado izquierdo de la página y el comienzo de la barra. Recuerde que la barra

está localizada en una caja que se encuentra centrada en la ventana. Los valores dependerán de cada situación en particular, por lo tanto, supongamos que la barra está localizada a 421 pixeles del lado izquierdo de la página web y el clic fue realizado en el medio de la barra.

Debido a que la barra tiene una longitud de 600 pixeles, el clic fue hecho a 300 pixeles desde el comienzo de la barra. Sin embargo, la propiedad **pageX** no retornará el valor 300, sino 721. Para obtener la posición exacta en la barra donde el clic ocurrió, debemos substraer de **pageX** la distancia desde el lado izquierdo de la página hasta el comienzo de la barra (en nuestro ejemplo, 421 pixeles). Esta distancia puede ser obtenida mediante la propiedad **offsetLeft**. Entonces, usando la formula **e.pageX - barra.offsetLeft** conseguimos exactamente la posición del puntero del ratón relativa al comienzo de la barra. En nuestro ejemplo, la fórmula en números sería: **721 – 421 = 300**.

Una vez obtenido este valor, debemos convertirlo a segundos. Usando la propiedad **duration**, la posición exacta del puntero del ratón en la barra y el tamaño máximo de la barra construimos la fórmula **ratonX × video.duration / maximo** y almacenamos el resultado dentro de la variable **nuevoTiempo**. Este resultado es el tiempo en segundos que la posición del puntero del ratón representa en la línea de tiempo.

El siguiente paso es comenzar a reproducir el video desde la nueva posición. La propiedad **currentTime**, como ya mencionamos, retorna la posición actual del video en segundos, pero también avanza o retrocede el video a un tiempo específico si un nuevo valor le es asignado. Con el código **medio.currentTime=nuevoTiempo** movemos el video a la posición deseada.

Lo único que resta por hacer es cambiar el tamaño del elemento **progreso** para reflejar en pantalla la nueva situación. Utilizando el valor de la variable **ratonX** cambiamos el tamaño del elemento para alcanzar exactamente la posición done el clic fue hecho.

El código para nuestro reproductor de video ya está casi listo. Tenemos todos los eventos, métodos, propiedades y funciones que nuestra aplicación necesita. Solo hay una cosa más que debemos hacer, un evento más que debemos escuchar para poner nuestro código en marcha:

```
window.addEventListener('load', iniciar, false);
```
***Listado 5-10.*** *Escuchando al evento* `load`.

Podríamos haber usado la técnica **window.onload** para registrar el manejador del evento, y de hecho hubiese sido la mejor opción para hacer nuestros códigos compatibles con viejos navegadores. Sin embargo, debido a que este libro es acerca de HTML5, decidimos usar el nuevo estándar **addEventListener()**.

**Hágalo usted mismo:** Copie todos los códigos Javascript desde el Listado 5-6 dentro del archivo **reproductor.js**. Abra el archivo **video.html** con la plantilla del Listado 5-4 en su navegador y haga clic en el botón “Reproducir”. Intente utilizar la aplicación desde diferentes navegadores.