CONTENIDO MULTIMEDIA HTML5

Usando audio y video con HTML5

HTML5 introduce soporte integrado para el contenido multimedia gracias a los elementos <audio> y <video>, ofreciendo la posibilidad de insertar contenido multimedia en documentos HTML.

Insertando contenido multimedia

Insertar contenido multimedia en tus documentos HTML es muy sencillo:

```
<video src="http://v2v.cc/~j/theora_testsuite/320x240.ogg" controls>
 Tu navegador no implementa el elemento <code>video</code>.
</video>
```

Este ejemplo reproduce un vídeo de ejemplo, con los controles de reproducción, desde el sitio Web de Theora.

Este es un ejemplo para insertar audio en tu documento HTML

```
<audio src="/test/audio.ogg">
<p>Tu navegador no implementa el elemento audio.</p>
</audio>
```

El atributo src puede ser una URL del archivo de audio o la ruta al archivo en el sistema local.

```
<audio src="audio.ogg" controls autoplay loop>
<p>Tu navegador no implementa el elemento audio</p>
</audio>
```

Este código de ejemplo usa los atributos del elemento <audio>:

controls : muestra los controles estándar de HTML5 para audio en una página web.

autoplay : hace que el audio se reproduzca automáticamente.

loop : hace que el audio se repita automáticamente.

```
<audio src="audio.mp3" preload="auto" controls></audio>
```

El atributo preload es usado en el elemento audio para almacenar temporalmente (buffering) archivos de gran tamaño. Este puede tomar uno de 3 valores:

"none" no almacena temporalmente el archivo

"auto" almacena temporalmente el archivo multimedia

"metadata" almacena temporalmente sólo los metadatos del archivo

Se pueden especificar múltiples fuentes de archivos usando el elemento <source> con el fin de proporcionar vídeo o audio codificados en formatos diferentes para diferentes navegadores. Por ejemplo:

```
<video controls>
 <source src="foo.ogg" type="video/ogg">
 <source src="foo.mp4" type="video/mp4">
 Tu navegador no implementa el elemento <code>video</code>.
</video>
```

Esto reproduce el archivo Ogg en navegadores que admiten el formato Ogg. Si el navegador no admite Ogg, el navegador usará el archivo MPEG-4. Mira también la lista de los formatos multimedia admitidos por los elementos audio y video en los diferentes navegadores.

También puedes especificar qué codecs requiere el archivo multimedia; de esta forma el navegador tomará decisiones más inteligentes:

```
<video controls>
 <source src="foo.ogg" type="video/ogg; codecs=dirac, speex">
 Tu navegador no implementa el elemento <code>video</code>.
</video>
```

Aquí, especificamos que el vídeo usa los codecs Dirac y Speex. Si el navegador implementa Ogg, pero no los codecs especificados, el vídeo no será cargado.

Si el atributo type no está especificado, el tipo de contenido multimedia se obtiene del servidor y se comprueba para ver si el navegador lo puede manejar; si no puede ser mostrado, se comprueba el siguiente source , si ninguno de los elementos source especificados pueden ser usados, un evento de error es enviado al elemento video. Si el atributo type está especificado, es comparado con los tipos que el navegador puede reproducir, y si no es reconocido, no se hace la consulta al servidor; en su lugar, el siguiente source se comprueba una vez.

Mira los eventos del contenido multimedia para una lista completa de eventos asociados con la producción multimedia. Para detalles en los formatos multimedia soportados por los diferentes navegadores, mira los formatos multimedia soportados por los elementos audio y video.

Controlando la reproducción multimedia

Una vez que has incrustado el contenido multimedia en tu documento HTML usando los nuevos elementos, puedes controlarlos mediante programación en JavaScript. Por ejemplo, para iniciar (o reiniciar) la reproducción, puedes hacer esto:

```
var v = document.getElementsByTagName("video")[0];
v.play();
```

La primera línea obtiene el primer elemento video en el documento, y la segunda línea llama al método play() del elemento, como está definido en la interfaz nsIDOMHTMLMediaElement que es usada para implementar los elementos multimedia.

Controlar un reproductor de audio en HTML5 para reproducir, pausar, aumentar y disminuir el volumen usando algo de Javascript es muy sencillo.

```
<audio id="demo" src="audio.mp3"></audio>
<div>
  <button onclick="document.getElementById('demo').play()">Reproducir el Audio</button>
  <button onclick="document.getElementById('demo').pause()">Pausar el Audio</button>
  <button onclick="document.getElementById('demo').volume+=0.1">Aumentar el Volumen</button>
  <button onclick="document.getElementById('demo').volume-=0.1">Disminuir el Volumen</button>
```

```
</div>
```

Deteniendo la descarga de contenido multimedia

Mientras que detener la reproducción multimedia es tan fácil como llamar al método pause() del elemento, el navegador sigue descargando el contenido multimedia hasta que el elemento multimedia es eliminado a través de la recolección de basura.

Aquí un truco para detener la descarga de una sola vez:

```
var mediaElement = document.getElementById("myMediaElementID");
mediaElement.pause();
mediaElement.src = "";
```

Estableciendo una cadena vacía al atributo src del elemento multimedia, tu destruyes el decodificador interno del elemento con lo que detienes la descarga.

Buscando a través de los medios

Los elementos de los medios proporcionan apoyo para mover la posición de reproducción actual a puntos específicos en el contenido de los medios. Esto se hace estableciendo el valor de la propiedad hora actual en el elemento; ver HTMLMediaElement para más detalles sobre las propiedades del elemento. Basta con establecer el valor en el tiempo, en segundos, con el que desea reproducir para continuar.

Usted puede utilizar el elemento seekable propiedad para determinar los rangos de los medios de comunicación que están disponibles para la búsqueda de la actualidad. Esto devuelve un objeto TimeRanges que enumera los rangos de veces que se puede tratar de:

```
var mediaElement = document.getElementById('mediaElementID');
mediaElement.seekable.start();  // Returns the starting time (in seconds)
mediaElement.seekable.end();    // Returns the ending time (in seconds)
mediaElement.currentTime = 122; // Seek to 122 seconds
mediaElement.played.end();      // Returns the number of seconds the browser has played
```

Especificación del rango de reproducción

Al especificar el URI de los medios de comunicación para un elemento <audio> o <video> , puede incluir opcionalmente información adicional para especificar la parte de los medios a reproducir. Para ello, añada una almohadilla ("#"), seguida de la descripción del fragmento de medios.

Un intervalo de tiempo se especifica mediante la sintaxis:

#t=[starttime][,endtime]

El tiempo se puede especificar como un número de segundos (como un valor de punto flotante) o como una hora / minuto / segundo tiempo separado con dos puntos (por ejemplo, 2:05:01 durante 2 horas, 5 minutos y 1 segundo).

Algunos ejemplos:

http://foo.com/video.ogg # t = 10,20

Especifica que el vídeo debe desempeñar el rango de 10 segundos a través de 20 segundos.

http://foo.com/video.ogg # t =, 10.5

Especifica que el vídeo se reproducirá desde el principio a través de 10,5 segundos.

http://foo.com/video.ogg # t =, 02:00:00

Especifica que el vídeo se reproducirá desde el principio a través de dos horas.

http://foo.com/video.ogg # t = 60

Especifica que el vídeo se debe reproducir desde los 60 segundos hasta el final.

Gecko 9.0 note

(Firefox 9.0 / Thunderbird 9.0 / SeaMonkey 2.6)

La porción rango de reproducción del elemento de la especificación URI media esta en Gecko 9.0 (Firefox 9.0 / Thunderbird 9.0 / SeaMonkey 2.6).

En este momento, esta es la única parte de la especificación de los medios de comunicación Fragmentos URIimplementado por el Gecko, y sólo se puede utilizar cuando se especifica la fuente para los elementos de los medios de comunicación, y no en la barra de direcciones.

Opciones de reserva

HTML incluido entre, por ejemplo, las etiquetas de apertura y cierre de los elementos de los medios de comunicación son procesados por los navegadores que no admitan medios de HTML5. Usted puede tomar ventaja de este hecho para proporcionar medios alternativos de reserva para esos navegadores.

Esta sección proporciona dos opciones de reserva para video. En cada caso, si el navegador soporta vídeo HTML5, que se utiliza, de lo contrario, se utiliza la opción de reserva.

Utilización de Flash

Puede usar Flash para reproducir una película formato Flash si el <video> no se admite elemento:

```
<video src="video.ogv" controls>
  <object data="flvplayer.swf" type="application/x-shockwave-flash">
    <param value="flvplayer.swf" name="movie"/>
  </object>
</video>
```

Tenga en cuenta que no se debe incluir classid en el objeto tag con el fin de ser compatible con los navegadores que no sean Internet Explorer.

Reproducción de vídeos Ogg utilizando un applet de Java

Hay un applet de Java llamada Cortado que se puede utilizar como reserva para reproducir vídeos Ogg en los navegadores que no tienen soporte para Java, pero no es compatible con vídeo HTML5:

```
<video src="my_ogg_video.ogg" controls width="320" height="240">
 <object type="application/x-java-applet" width="320" height="240">
   <param name="archive" value="cortado.jar">
   <param name="code" value="com.fluendo.player.Cortado.class">
   <param name="url" value="my_ogg_video.ogg">
   <p>You need to install Java to play this file.</p>
 </object>
</video>
```

Si no se crea un elemento secundario alternativa del elemento de objeto cortado, como el <p> elemento superior, Firefox 3.5 instalaciones que manejan el video de forma nativa, pero no tienen Java instalado incorrectamente informarán al usuario de que es necesario instalar un plugin para ver el contenido en la página.

(Firefox 4 / Thunderbird 3.3 / SeaMonkey 2.1)

Gestión de errores

A partir de Gecko 2.0 (Firefox 4 / Thunderbird 3.3 / SeaMonkey 2.1), el tratamiento de errores se ha revisado para que coincida con la última versión de la especificación HTML5. En lugar de que el error sea enviado al elemento en sí, ahora se entrega a los elementos "hijos" <source> correspondientes a las fuentes que resultan en el error.

Esto permite detectar las fuentes no pudieron cargar, que puede ser útil. Considere este HTML:

```
<video>
<source id="mp4_src"
```

```
 src="video.mp4"
 type='video/mp4; codecs="avc1.42E01E, mp4a.40.2"'>
</source>
<source id="3gp_src"
 src="video.3gp"
 type='video/3gpp; codecs="mp4v.20.8, samr"'>
</source>
<source id="ogg_src"
 src="video.ogv"
 type='video/ogg; codecs="theora, vorbis"'>
</source>
</video>
```

Dado que Firefox no es compatible con MP4 y 3GP, debido a su naturaleza de patente gravado, los <source> elementos con el ID "mp4_src" y "3gp_src" recibirán error eventos antes de cargar el recurso Ogg. Las fuentes son juzgados en el orden en el que aparecen, y una vez que uno carga con éxito, las fuentes restantes no se trataron en absoluto.

Detectar si las fuentes no han cargado

Para detectar qué todos los elementos <source> no han podido cargarse, se debe comprobar el valor de la propiedad NetworkState qué poseen todos los elementos de tipo multimedia. Si el valor es HTMLMediaElement.NETWORK_NO_SOURCE, se sabrá que las fuentes no se cargaron correctamente.

Si en ese momento se agrega otra fuente mediante la inserción de un nuevo elemento <source> como hijo del elemento multimedia, Gecko intenta cargar el recurso especificado.

Mostrando contenido fallback cuando la fuente no puede ser cargada

Otra forma de mostrar el contenido fallback de un vídeo cuando ninguna de sus fuentes pudieron ser cargadas, es añadir un manejador de excepciones o errores en el último elemento <source>. Así usted podrá reemplazar el vídeo con el contenido fallback:

```
<video controls>
 <source src="dynamicsearch.mp4" type="video/mp4"></source>
 <a href="dynamicsearch.mp4">
  <img src="dynamicsearch.jpg" alt="App de b&uaacute;squeda din&aacute;mica en Firefox OS">
 </a>
 <p>Clic en la im&aacute;gen para reproducir un v&iacute;deo demo de la app de b&uaacute;squeda din&aacute;mica</p>
</video>
var v = document.querySelector('video'),
  sources = v.querySelectorAll('source'),
  lastsource = sources[sources.length-1];
lastsource.addEventListener('error', function(ev) {
 var d = document.createElement('div');
 d.innerHTML = v.innerHTML;
 v.parentNode.replaceChild(d, v);
}, false);
```