JESUÏTES
educació

DAWM12
GP1 FRAMEWORKS AVANÇATS EN JS

GP1 6.2 HTML5 AUDIO/VIDEO

CFGS Desenvolupament d'Aplicacions Web
M12. Projecte Final - Global Project
Fundació Jesuïtes Educació - Escola del Clot
Sergi Grau sergi.grau@fje.edu

OBJECTIUS

+ Aprendre l’APId’audio i video

INTRODUCCIÓN

+ Con HTML5, el audio y el vídeo se han convertido en lo más importante de la Web, tal como ocurrió en el pasado con otro tipo de contenido multimedia, como las imágenes. Las nuevas API permiten manipular los estados de red y los datos cronológicos de los archivos, controlarlos y acceder a ellos. También hay una API que permite leer y escribir datos sin procesar en archivos de audio (API de datos de audio, Audio Data API) y para manipular subtítulos de vídeos (API de pistas temporizadas, Timed Track API). Sin embargo, las auténticas posibilidades de estos nuevos elementos HTML se descubren al combinarlos con otras tecnologías existentes en la Web, como el elemento canvas, la especificación SVG, el lenguaje CSS e incluso la especificación WebGL.

VERIFICANT EL NAVEGADOR

- http://www.whatwg.org/specs/web-apps/current-work/multipage/the-video-element.html

- http://www.w3.org/TR/html5/embedded-content-0.html#the-video-element

- http://caniuse.com/

HTML5 VIDEO

- En los navegadores modernos, la adición de un vídeo es tan fácil como añadir una imagen. Ya no se necesitan los plug-ins especiales. Todo lo que necesita para incrustar un simple video en una página y mostrar los controles básicos para que un usuario puede reproducir, pausar o controlar de otro modo el vídeo se puede apreciar en este ejemplo.

```
<video src="video.webm" controls>
```

HTML5 VIDEO

- Puede especificar varios archivos de código fuente utilizando el <source> elemento. El elemento de origen le permite especificar múltiples formatos como reserva en caso de que el navegador del usuario no admite uno de ellos. Por ejemplo:

```
<video controls> <source src="devstories.webm"
type='video/webm;codecs="vp8, vorbis"'/> <source
src="devstories.mp4" type='video/
mp4;codecs="avc1.42E01E, mp4a.40.2"'/> </video>
```

HTML5 AUDIO

- Este es un ejemplo para inscrustar 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>`

HTML5 AUDIO: ATRIBUTOS

- autoplay, Un atributo booleano; si se especifica (incluso aunque el valor sea "false"), el sonido comenzará a reproducirse automáticamente en cuanto sea posible, sin detenerse para terminar de cargar los datos.

- autobuffer, Un atributo booleano; si se especifica, el sonido comenzará a reproducirse automáticamente, incluso aunque no se haya configurado para la reproducción automática. Esto continuará hasta que la caché de medios esté llena o se haya descargado el archivo de audio completo, lo que suceda primero. Debería usarse sólo si se espera que el usuario elija reproducir el audio; por ejemplo si el usuario ha navegado hasta una página usando un vínculo de "Reproducir este audio". Este atributo se eliminó de Gecko 2.0 (Firefox 4 / Thunderbird 3.3 / SeaMonkey 2.1) en favor del atributo preload.

- buffered, Un atributo que se puede leer para determinar qué intervalos de tiempo del multimedia se han almacenado en búfer. Este atributo contiene un objeto TimeRanges.

- controls, Si está presente este atributo, el navegador ofrecerá controles para permitir que el usuario controle la reproducción de audio, incluyendo volumen, búsqueda y pausar/reanudar reproducción.

HTML5 AUDIO: ATRIBUTOS

- loop , Un atributo booleano; si se especifica, al alcanzar el final del audio, realizaremos la búsqueda automáticamente hasta el principio.

- preload , El objetivo de este atributo enumerado es proporcionar una sugerencia al navegador sobre qué cree el autor que proporcionará la mejor experiencia para el usuario . Puede tener uno de los siguientes valores:

- none: sugiere bien que el autor cree que el usuario no tendrá que consultar ese video, bien que el servidor desea minimizar su tráfico; es decir, esta sugerencia indica que no se debe almacenar en caché este video;

- metadata: sugiere que aunque el autor piensa que el usuario no tendrá que consultar ese video, es razonable capturar los metadatos (p. ej. longitud);

- auto: sugiere que el usuario necesita tener prioridad; es decir, esta sugerencia indica que, si es necesario, se puede descargar el video completo, incluso aunque el usuario no vaya a usarlo;

- the empty string: que es sinónimo del valor auto.

- Si no está configurado, su valor predeterminado está definido por el navegador (es decir, cada navegador puede elegir su propio valor predeterminado), aunque la especificación aconseje que

HTML5 AUDIO: ATRIBUTOS

- src, La URL del audio que se va a insertar. Está sujeta a los Controles de acceso HTTP. Es opcional; en su lugar puedes usar el elemento source dentro del bloque de audio para especificar el audio que se va a insertar.

- Las compensaciones de tiempo se especifican como valores float que indican el número de segundos que se va a compensar.

MULTIPLES ORÍGENES

- 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>
```

CONTROLANDO LA REPRODUCCIÓN MULTIMEDIA

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

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

CONTROLANDO LA REPRODUCCIÓN 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.

```
let mediaElement = document.getElementById(“elemento");
mediaElement.pause();
mediaElement.src = "";
```

CONTROLANDO LA REPRODUCCIÓN MULTIMEDIA

```
<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>
```

BUSCANDO A TRAVÉS DE LOS MEDIOS DE COMUNICACIÓN

Elementos de los medios de comunicación 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 horaActual propiedad en el elemento; ver HTMLMediaElementpara más detalles sobre las propiedades del elemento. Basta con establecer el valor en el tiempo, en segundos, con el que desea reproducir para continuar.

```
let mediaElement = document.getElementById('mediaElementID');
mediaElement.seekable.start();  // temps inici en segons
mediaElement.seekable.end();    // temps final en segons
mediaElement.currentTime = 122; // salta a 122s
mediaElement.played.end();      // nombre de segons reproduïts
```

HTML5 VIDEO

- Para mejorar el rendimiento, siempre se debe incluir el atributo type en el elemento source. De lo contrario, el navegador tendrá que cargar tipo de codec hasta que encuentre uno con el que reproducirlo.

- Lo más importante que se debe recordar es que tienes que asegurarte de que el servidor muestre los archivos de vídeo con el tipo MIME correcto en el encabezado Content-Type. De lo contrario, los vídeos podrían no funcionar correctamente (aunque sí funcionen en una copia local del sitio). En un archivo de configuración "httpd.conf" de Apache, solo habría que añadir estas líneas:

```
AddType video/ogg .ogv
AddType video/mp4 .mp4
AddType video/webm .webm
```

FORMATOS DE VÍDEO

El concepto de formato de vídeo se puede entender como un archivo zip que contiene secuencias de vídeo y secuencias de audio. Los tres formatos que se deben tener en cuenta para la Web son WebM, MP4 y OGV:

.mp4 = H.264 + AAC
.ogg/.ogv = Theora + Vorbis
.webm = VP8 + Vorbis

https://caniuse.com/?search=video%20format

CONTROLANDO LA REPRODUCCIÓN MULTIMEDIA


```
<!DOCTYPE html>
<html lang="ca">
    <head>
        <title>Exercici14</title>
        <meta charset="utf-8" />
    </head>
    <body>

        <video controls style="width:640px;height:360px;"
poster="logo.png">
        <source src="big_buck_bunny.webm"
        type='video/webm;codecs="vp8, vorbis"' />
        <source src="big_buck_bunny.mp4"
        type='video/mp4;codecs="avc1.42E01E, mp4a.40.2"' />
        <source src="big_buck_bunny.ogv" type='video/ogg; codecs="theora,
vorbis"'/>
        Navegador no suporta video HTML5

        </video>

    </body>
</html>
```

ATRIBUTOS

- Todos los atributos HTML comunes se pueden utilizar ahora en el reproductor de vídeo. Por ejemplo, en el fragmento que aparece a continuación, se utiliza tabindex para poder acceder a los controles del reproductor. Hay algunos atributos nuevos que se pueden utilizar en la etiqueta de vídeo que también son comunes a la etiqueta de audio, como el atributo de reproducción continua (loop) y el de reproducción automática (autoplay). El atributo poster indica qué imagen se mostrará durante la carga inicial del vídeo y al final de su reproducción, mientras que controls se utiliza para indicar que, en lugar de crear controles personalizados, queremos que el navegador muestre los controles automáticamente. También hay un atributo preload, que se puede utilizar para descargar el vídeo en segundo plano en cuanto se carga la página aunque no haya empezado a reproducirse.

```
<video poster="star.png" autoplay loop controls tabindex="0">
  <source src="movie.webm" type='video/webm; codecs="vp8,
vorbis"' />
  <source src="movie.ogv" type='video/ogg; codecs="theora, vorbis"'
/>
</video>
```

- Como podrás imaginarte, se puede aplicar estilo a la etiqueta de vídeo con propiedades CSS tradicionales (p. ej., bordes, opacidad, etc.), ya que esta etiqueta es uno de los elementos principales del DOM, pero lo mejor es que también se le pueden aplicar las últimas propiedades CSS3, como reflejos, máscaras, gradientes, transformaciones, transiciones y animaciones.

VIDEO + CSS

```
video {
            opacity:0.75;
            position: absolute;
            top: 50%;
            left: 50%;
            width: 120px;
            height: 120px;
            margin: -60px 0 0 -60px;
            -webkit-animation: gira 4s linear infinite;
            -moz-animation: gira 4s linear infinite;
            animation: gira 4s linear infinite;
        }
        @-moz-keyframes gira { 100% { -moz-transform: rotate(360deg); } }
        @-webkit-keyframes gira { 100% { -webkit-transform:
rotate(360deg); } }
        @keyframes gira { 100% { -webkit-transform: rotate(360deg);
transform:rotate(360deg); } }
```

VIDEO + CANVAS

- El elemento canvas es otro elemento HTML5 que ofrece muchas posibilidades cuando se utiliza junto con la etiqueta de vídeo.

- Exemple github.com/sergigrau

VIDEO + SVG

+ SVG permite mostrar y manipular gráficos vectoriales mediante programación, pero también incluye más funciones, como los efectos de filtro SVG. Estos filtros permiten elegir un elemento DOM específico y aplicarle algunos efectos especiales como los de mosaicos, difuminado, composición.

```
<svg id='image' version="1.1" xmlns="http://www.w3.org/
2000/svg">
  <defs>
    <filter id="filtre">
      <feGaussianBlur stdDeviation="1" />
    </filter>
  </defs>
</svg>
<style>
  video { filter:url(#filtre); border: 2px solid red; }
</style>
```

FRAGMENTOS DE MEDIOS

- Con la adición de un fragmento de los medios a URL, se puede especificar la porción exacta que se desea reproducir . Para agregar un fragmento de los medios, sólo se tiene que añadir #t=[start_time][,end_time] a la URL . Por ejemplo, para reproducir el vídeo entre el segundo 10 a 20, puede especificar:

```
<source src="devstories.webm#t=10,20"
        type='video/webm;codecs="vp8, vorbis"' />
```

- También puede especificar el tiempo en hours:minutes:seconds , como #t=00:01:05 para iniciar el vídeo en un minuto, cinco segundos

SUBTÍTULOS

- El elemento <track> proporciona una manera simple y estandarizada para añadir subtítulos, títulos, descripciones de lectores de pantalla y los capítulos a un video, lo que mejora la accesibilidad, sino también hace posible que los motores de búsqueda para entender lo que está en el video. Además de los subtítulos y los subtítulos, es posible poner los metadatos en las señales, por ejemplo, en formato JSON.

```
<video controls style="width:640px;height:360px;"
poster="poster.png">
  <source src="devstories.webm"
          type='video/webm;codecs="vp8, vorbis"' />
  <source src="devstories.mp4"
          type='video/mp4;codecs="avc1.42E01E, mp4a.40.2"'
/>
  <track src="subtitols-en.vtt" label="English subtitles"
         kind="subtitles" srclang="en" default></track>
</video>
```

+ Los <track> van dentro del elemento <video>, y tiene un atributo src que apunta a un archivo en formato WebVTT. Puede especificar la etiqueta que se mostrará en la interfaz de usuario para el usuario, así como el idioma de origen (srclang) y si hay varios, cuál debe ser usado por defecto.

```
WEBVTT FILE

1
00:00:00.500 --> 00:00:02.000 D:vertical A:start
The Web is always changing

2
00:00:02.500 --> 00:00:04.300
and the way we access it is changing

3
00:00:05.000 --> 00:00:07.000
The source of that change is <c.highlight>you</c>
```

Properties

`currentTime`	Gets or sets the current playback position in seconds
`volume`	Gets or sets the current volume level for the video
`muted`	Gets or sets the mute state
`playbackRate`	Gets or sets the playback rate, where 1 is normal speed forward
`currentSrc`	Returns the current video source file the browser is playing
`videoWidth` & `videoHeight`	Returns the actual dimensions of the video, not the video element size

Methods

`load()`	Loads the video and reset the play head to the beginning of the video
`play()`	Plays the video from it's current location
`pause()`	Pauses the video at the current location
`canPlayType(format)`	Tests to see whether the browser can play a specific type of video, for example `'video/webm;codecs="vp8, vorbis"'` The browser will return: • **`probably`** - if it's most likely the video file can be played • **`maybe`** - if the video might be playable • **`[empty string]`** - if the video file is not playable

Events*

`canplaythrough`	Fired when enough data is available that the browser believes it can play the video completely without interruption
`ended`	Fired when the video has finished playing
`error`	Fired if an error occurs
`playing`	Fired when the video starts playing, for the first time, after being paused or when restarting
`progress`	Fired periodically to indicate the progress of downloading the video
`waiting`	Fired when an action is delayed pending the completion of another action
`loadedmetadata`	Fired when the browser has finished loading the metadata for the video and all attributes have been populated

BIBLIOGRAFIA

+ https://caniuse.com/?search=video%20format

+ https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Client-side_web_APIs/Video_and_audio_APIs