IT614

Audio and Video in HTML5

HTML5 introduced built-in media support via the <audio> and <video> elements, offering the ability to easily embed media into HTML documents. HTML5 can play audio and video files without plugins. A good introduction is here: https://developer.mozilla.org/en-US/docs/Web/Guide/HTML/Using_HTML5_audio_and_video.

1. Codecs and file formats

A codec, short for encoder/decoder, is a compression standard. Raw video or audio is compressed when encoding, and decompressed (decoded) on playback. The player would know how to decompress the file.

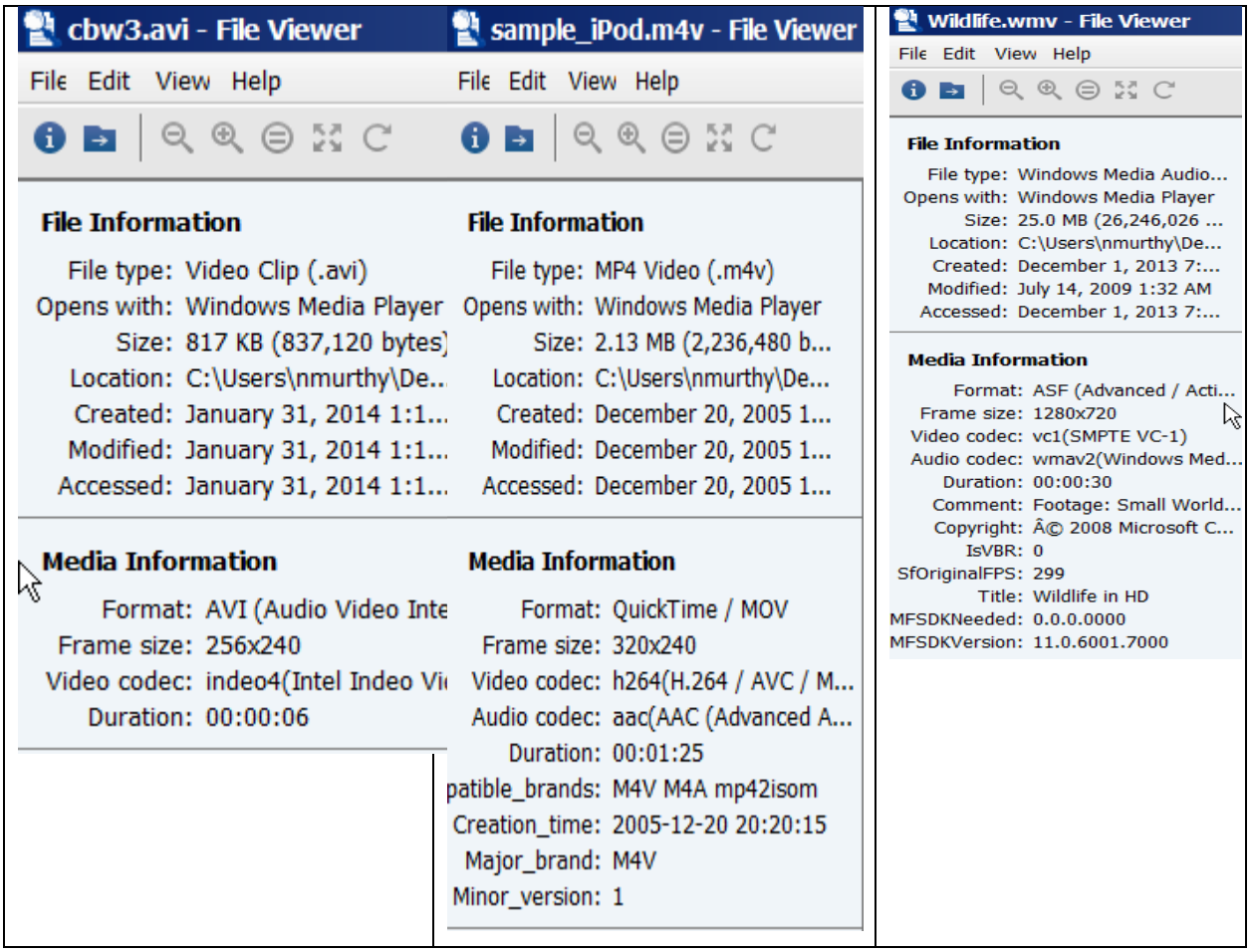
There are hundreds of codecs out there; some of the more important ones are MPEG-2, H.264, Theora, VP8, DivX, XviD, and the WMV family (video) and MP3, AAC, Vorbis, and the WMA family (audio).

A media file a container that holds one or more video or audio (or both) codecs. The container describes the structure of the file: where the various pieces are stored, how they are made (interleaved), and which codecs are used by which pieces. It is something that carries all of the information that tells the computer how to read the file.

Some of the commonly used container formats are mov (Quicktime), mp4, ogg, and avi. The container format defines how the metadata is stored along with the audio and video data. A container may hold more than one codec.

Using a file viewer

You can use a file viewer program to view the metadata in a file. I use a viewer downloaded from http://windowsfileviewer.com/. Here are three examples of metadata:



2. Audio and video formats supported by HTML5

The HTML5 <audio> tag supports three audio formats: MP3, Wav, and Ogg.
The HTML5 <video> tag supports three video formats: MP4, WebM, and Ogg.
It does not mean all browsers support all the formats. You have to experiment with your browser and determine if your browser supports these.

WebM is a new open standard for compressed Video Content. It is an alternative to commercial formats such as h264, MPEG4 and MPEG2 that are patented and require a commercial license. WebM is fully supported by Google and Firefox. IE does not (yet) support WebM. It is hard to find WebM files on the Web. Try these Web sites: http://www.webmfiles.org/demo-files/, http://techslides.com/demos/sample-videos/small.webm, http://podhawk.com/index.php?cat=samples, and http://www.ioncannon.net/examples/vp8-webm/demo.html.

3. The <audio> tag

The <audio>…</audio> tag is used to include audio on an HTML page. This tag is new in HTML5. HTML5 supports three kinds of sound files: mp3, wav and ogg. Chrome supports all the three formats. If you are using a different browser, you have to try.

Syntax:

<**audio src**="*audioFileLocation*" **controls**>
text
</audio>

The text between <audio> and </audio> will not be visible if the browser supports the embedded audio file. Otherwise, the text shows up. You would include a warning that "the audio is not supported in this browser."

The controls attribute is required to show the audio controls:

Important attributes:

Attribute	Value	Description
autoplay	None	This boolean attribute if specified, the audio will automatically begin to play back as soon as the file is loaded
loop	None	Loops the audio file – keeps on playing.
controls	None	Displays the standard HTML5 controls for the audio on the web page.

4. The <video> tag

The <video>…</video> tag is used to include video on an HTML page. This tag is new in HTML5. HTML5 supports three kinds of video files: mp4, WebM and ogg. Chrome supports all the three formats. If you are using a different browser, you have to try.

Syntax:

<**video src**="*videoFileLocation*" **width**=*w* **height**=*h* **controls**>
text
</video>

The text between <video> and </video> will not be visible if the browser supports the embedded video file. Otherwise, the text shows up. You would include a warning that "the video is not supported in this browser." The values *w* and *h* (integers specifying pixels) of attributes **width** and **height** specify width and the height of the video display screen.

Example:

```
<video src="podcast-2012-02-02-56559.webm" width=300 height=250 controls>
  Your browser does not support the video tag.
</video>
```

Important attributes:

Attribute	Value	Description
autoplay	None	This boolean attribute if specified, the video will automatically begin to play back as soon as the file is loaded
loop	None	Loops the audio file – keeps on playing.
controls	None	Displays the standard HTML5 controls for the video on the web page.

You can include your own controls

Once you've embedded media into your HTML document using <audio> and <video>, you can programmatically control them from simple JavaScript code. The following example shows how easy it is to have your own play, pause, increase and decrease volume controls using simple JavaScript.

```
<head>
<script type="text/javascript"></script>
</head>
<body>
<video src="Walmart.mp4" id="videoExample" width=300 height=350></video>
<div>
  <button onclick="document.getElementById('videoExample').play()">Play the video</button>
  <button onclick="document.getElementById('videoExample').pause()">Pause the video</button>
  <button onclick="document.getElementById('videoExample').volume+=0.1">Increase Volume</button>
  <button onclick="document.getElementById('videoExample').volume-=0.1">Decrease Volume</button>
</div>
</body>
```

Similar code will work for audio controls.

5. Placing video behind HTML

This is an interesting example. There are plenty of examples on this topic on the Web. This example plays a video clip in the background as the user reads the regular HTML on the page.

```
<!DOCTYPE html>
<html>
 <head> <script src="jquery.js"></script>
 <head>
 <style>
 @font-face {
 font-family: nmfont; src: url(Tangerine_Bold.ttf);
 }
#videobg {
position: absolute; bottom: 0px; right: 0px; min-width: 100%;
min-height: 100%; width: auto; height: auto; z-index: -1000;
overflow: hidden; }

h3{
border-style:solid;
border-color:blue;
padding:20px;
width:300px;
height:430px;
color:red;
}
h2{
font-family: nmfont;
font-size:52px;
text-align : center;
color:white;
margin-top:150px;
}
</style>
</head>
<body >
<video id="videobg" preload="auto" autoplay="true" loop="loop"  volume="0" src="Heroes-
TheTankman.mp4" type="video/mp4"> </video>
```

```
<div><b><h3><i>The incident took place near Tiananmen on June 4th, 1989,
one day after the Chinese government's violent crackdown on the Tiananmen protests.
The man stood in the middle of the wide avenue, directly in the path of a column of approaching Type 59
tanks.
 In response, the lead tank attempted to drive around the man, but the man
repeatedly stepped into the path of the tank in a show of nonviolent action.
After repeatedly attempting to go around rather than crush the man, the lead tank stopped
its engines, and the armored vehicles behind it seemed to follow suit. There was a short pause with
the man and the tanks having reached a quiet, still impasse.</i></h3></b></div>
<h2>A hero is someone who is ready to give his life to something bigger than oneself.</h2>
</body>
</html>
```

Try it: http://vulcan.seidenberg.pace.edu/~nmurthy/IT614/Chapter9/BackgroundVideo.html
(Works well in Chrome.)

6. More media element properties and methods

We have used play(), and pause() methods of a media element. The media element provides several other methods and properties. We will see a couple more. For a complete list see: https://developer.mozilla.org/en-US/docs/Web/API/HTMLMediaElement.

➢ The playbackRate property

The playbackRate property holds the current rate at which the media is being played back. The value of playbackRate is a number and for normal speed this value is 1.0

Example:

```
<!DOCTYPE html>
<html>
 <head> <script src="jquery.js"></script>
 <head>
 <script>
 function slower(){
  var myVideo = document.getElementById('demo');
  myVideo.playbackRate = myVideo.playbackRate-0.25;
  }
  function faster(){
  var myVideo = document.getElementById('demo');
  myVideo.playbackRate = myVideo.playbackRate+0.25;
```

```
 }
</script>
</head>
<body >
<video src="redcliff450.webm" autoplay loop  id="demo"  type="video/webm"></video><br>
<button onclick='slower()'>Slower</button>
<button onclick='faster()'>Faster</button>
</body>
</html>
```

Every time you press the slower button, the value of playbackRate is reduced by 0.25 in the slower() function. Similarly the Faster button makes it run faster.

Try it: http://vulcan.seidenberg.pace.edu/~nmurthy/IT614/Chapter9/SlowFast.html

Note:

If the playbackRate is negative, the media is played backwards. But this is not yet implemented in browsers.

7. Playing a portion of the media

You can specify a portion of the media to be played using the following syntax:

#t=s,e

Where **#t** is required, s is the starting point and e is the ending point. *s* and *e* are numbers indicating the starting time and ending time within the duration of the media.

Examples:

<video src="redcliff450.webm#t=40,50" autoplay controls></video>

This plays from 40th second through 50th second of the video.

<video src="redcliff450.webm#t=,30" autoplay controls></video>

This plays the first 30 seconds of the video.

<video src="redcliff450.webm#t=55" autoplay controls></video>

This plays starting at 55th second through the end of the video.

8. The <track> element

The <track> tag used within <audio></audio> and <video></video> elements to include subtitles, captions, and other descriptions to video and audio. <track> is a void element – no end tag.

We will discuss <track> within <video></video>. Similar ideas hold good for <audio></audio>.

Syntax:

<**video** ….>
  <**track** *attributes*>
</**video**>

<track> attributes:

We will discuss only one feature - subtitles. For more information, see http://www.w3.org/TR/html5/embedded-content-0.html#the-track-element

attribute	Values and Description
kind	value: **subtitles** Use this if you like to provided subtitles for your video. Instead of actual subtitles, you can provide any kind of information about the video the user is watching. Subtitles is the default value.
src	Value is the file name which describes the "cues". Details below.
srclang	Language of the track text data. If the **kind** attribute is set to subtitles, then **srclang** must be defined.
label	A user-readable title of the text track which is used by the browser when listing available text tracks.

Note:

Normally you include several <track> elements, one for each language. The video element is supposed to display this list with specified label values for the user to select subtitle language. But, unfortunately, browsers have not implemented this features yet.

More on the src attribute

The value of the src attribute is a file, called WebVTT file, which contains the subtitles to be displayed when the video is playing.

Details about the WevVTT file:

- WebVTT stands for The Web Video Text Tracks Format
- A WebVTT is a text file, created using any text editor

- The first line of this file should contain the text: WEBVTT
- A WebVTT file must be encoded in UTF-8 format (in Notepad++, click Encoding button and select UTF-8).
- The mime type of WebVTT is text/vtt.
- Save the file with the extension .vtt.
- A WebVTT file contains several items, called cues. Each cue describes subtitle.

The format of a cue:

A cue has three parts: cue identifier, cue timing and cue payload.

- **Cue identifier**

The identifier is a name that identifies the cue. It can be used to reference the cue from a script. It can be just a number. A cue identifier is optional.

- **Cue timing**

A cue timing indicates when the subtitle is to be displayed. It has a start and end times

Syntax for a cue:

startTime --> endTime

The *startTime* and the *endTime* are in the format mm:ss.ttt or hh:mm:ss.ttt

(hh- hours, mm-minues, ss-seconds and ttt-miliseconds).

- **Cue Payload**

This is the actual subtitle text. This can be multiple lines (but no blank line).

Example:

This is an example a cue.

one

00:00:10 --> 00:00:25

Welcome to my video. I hope you enjoy this.

Here one is the cue identifier. The text, "Welcome to my video. I hope you enjoy this." is the actual subtitle displayed starting at 00:00:10 and ending at 00:00:25 of the video play. Typically you will have several cues like this in your WebVTT file.

Important Note:

- Each of these cue identifier, cue timing and cue payload must be on a separate line.
- There must not be blank lines between them
- There must be a space on either side of --> (before and after the -->).

A complete example:

sample.vtt

```
WEBVTT

1
00:00:10.000 --> 00:00:17.000
Welcome to this wondeful video.  I hope you will like it.

2
00:00:18.200 --> 00:00:23.000
I downloaded the video from the Web site
http://video.webmfiles.org/big-buck-bunny_trailer.webm

3
00:00:23.500 --> 00:00:30.000
This is a WebM file. It seems WebM will be the format for youtube in the future.
```

subtitls.html

```
<!DOCTYPE html>
<html>
 <head> <title>subtitles Example</title></head>
<body>
<video src="big-buck-bunny_trailer.webm" controls>
   <track src="one.vtt" kind="subtitles" label="English" srclang="en" default >
</video><br>
</body>
</html>
```

Try it: http://vulcan.seidenberg.pace.edu/~nmurthy/IT614/Chapter9/subtitls.html

Important notes:

1. This works well in Chrome. It will work in IE. Firefox does not support it yet.
2. Notice the CC icon: 

. You can hide subtitles by clicking this icon.

3. One important point: This will not work if open the HTML file locally (from your computer). It is something to do with same-origin policy. Chrome gives an error message: `Cross-origin text track load denied by Cross-Origin Resource Sharing policy.` So you must upload it to a server for this to work. I uploaded the HTML and related files to Vulcan. It is working.
