The video element is one of the most exciting new facilities provided in HTML5, but sometimes our plans require extra attention to detail and, perhaps, patience. This article starts with a very brief background on video formats and then describes my approach to solving a specific problem involving a video clip taken with an iPhone.

A frequent occurrence for some of us is to acquire a video clip in the Quick Time (MOV) format. This happens to me fairly often because my granddaughter’s parents, my son, daughter-in-law and her two brothers, all own iPhones. The problem arises because the MOV format contains information about the orientation of the video: think of portrait versus landscape. If you attempt to view the MOV clip using the Quick Time player, there is no problem. However, when the MOV format is used as the source in a video conversion program such as Miro to produce mpeg, ogg, or webm formats, the orientation information is lost. These other formats are the ones accepted for use in the new HTML5 video element. Figure 1 shows the result of taking a MOV video clip and converting it to each of the three formats recognized by HTML5 and using the standard HTML code shown in Listing 1.

This is not the orientation shot by the photographer. There is a fairly easy way to rotate the image, though at this point in time, it requires use of browser-specific Cascading Style Sheet (CSS) libraries, which I will explain. Figure 2 shows my first attempt to rotate the video.

Listing 1: Standard HTML5 coding for a video element

```html
<!DOCTYPE html>
<html>
<head>
  <title>Opinion</title>
</head>
<body>
  <video id="vid" controls="controls" preload="auto">
    <source src="blrrr.mp4" type='video/mp4; codecs="avc1.42E01E, mp4a.40.2"'>
    <source src="blrrr.ogv" type='video/ogg; codecs="theora, vorbis"'>
    <source src="blrrr.webm" type="video/webm; codec="vp8, vorbis"">
    Your browser does not accept the video tag.
  </video>
</body>
</html>
```

Figure 1: Screen shot of Annika showing her opinion

Figure 2: Rotated video element
This is better, but the vertical control bar is just plain silly. My next attempt did away with the built-in controls and used a pair of buttons to produce a toggle button effect. Figure 3 shows a screen capture while the video clip is playing, with a Pause button to pause it.

![Figure 3: Playing video clip, showing Pause button](image)

I added a restart feature to the application, necessary because the loop property does not work in some browsers (e.g., Firefox). Similarly, I also added a feature to start the video immediately, but that only works in Firefox. Lastly, I set off Annika’s sweet face (even though she is making somewhat rude noises) with a rounded-corners border.

**Brief note on video formats**

This article is not intended to be an explanation of video encoding, but rather to help you appreciate the issue and understand why there are multiple formats. Video represents a considerable amount of information: 24 frames per second or faster, with each frame being an image. The challenge of video is to manage the trade-off of file size, costly in terms of storage space and times for downloading, versus high quality of the video. The different formats, also called *codecs* (for compress/decompress) handle the challenge in different ways. For example, the codecs attempt to detect areas of each video frame that are similar and also areas that are similar frame to frame. Please keep in mind that an intricate way of compressing video may require a computationally intensive decoding to display the video. If you have multiple versions of the same video clip in different formats, you may find it interesting to look at the file sizes. The file sizes for the original MOV file of Annika, along with the three files I created using Miro are shown in Table 1.

<table>
<thead>
<tr>
<th>Format</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpeg</td>
<td>5379</td>
</tr>
<tr>
<td>ogg</td>
<td>3249</td>
</tr>
<tr>
<td>webm</td>
<td>1839</td>
</tr>
<tr>
<td>mov</td>
<td>3272</td>
</tr>
</tbody>
</table>

**Table 1: Comparative sizes of video encodings**

At the start of development for HTML5, each browser seemed to recognize a different one of the first three formats, but the situation has improved. In terms of file size for this particular clip, it looks like webm is the way to go if you are using the latest Chrome, Firefox or Opera browser. However, the recommended practice still is to prepare mpeg, ogg, and webm versions of each clip. You can make the judgment of how the different formats compare in terms of quality, keeping in mind that device (desktop computer, laptop computer, tablet, phone) and operating system also are significant factors.

**Note:** The mpeg format is "encumbered with patent issues." You are welcome to do research on this matter. I must confess that it made my fairly well-educated (S.B. from University of Chicago and Ph.D. from NYU) brain ache.

If you have the QuickTime plug-in, you can view MOV clips on your computer, but these are not available using the video element in HTML5. The problem of restoring the desired orientation is independent of the decision to choose one of mpeg, ogg or webm, so let’s move on to that issue and a couple more.

![Figure 4: Paused video clip, showing Play button](image)
Transform feature

The task is to rotate the video clip a quarter turn, 90 degrees. The transform feature is part of the HTML5/CSS definition and its functions include performing rotations; however, it is not yet recognized by any browser. Instead, we can use browser specific libraries. Listing 2 shows a recommended way to use a style directive for the element with id="vid" to position and rotate an element. The positioning is necessary because the rotation would make part of the element off the screen. Notice that unlike many functions in JavaScript and other programming languages, the unit for rotation is degrees (not radians). Notice also that the coding includes calls to three libraries plus the built-in, but not yet recognized transform itself. Listing 2 produces what is shown in Figure 2.

Listing 2: The style directive for rotating and positioning the video element

Now, I will describe the rest of the application, namely the use of certain video events and creating the Play and Pause buttons.

Video events

I wanted to remove the now vertical control bar, but I felt the need for a way to start and pause the video. I also wanted the video to loop, that is, re-start when it had finished. HTML5 and JavaScript provide many events and methods associated with video elements along with the attributes that can be included in the HTML markup of the video tag. One event is canplaythrough. This is defined to signal when enough of the video has been loaded to ensure that it will play through without stopping. Therefore, I could write code to respond to this event by starting the clip. It turned out that this did not go well. The use of innerHTML, for example, looked tacky. So, I created two buttons. Only one button would appear at a time through the use of display directives in the style element, as shown in Listing 4 and JavaScript, as shown in Listing 5. Notice it is necessary to position the buttons explicitly to place them under the rotated video element.

Listing 3: The init and restart functions

I could have included the loop attribute in the video tag, but chose not to do so. There are many more video events, properties and methods, enough to make an elaborate video playback system. The next section describes my limited goals.

Toggled button(s)

My goal was to have one button toggle between displaying Play and displaying Pause. That is, the label would be Play when the clip was not playing, such as the situation at the start for the browsers that did not recognize the canplaythrough event and the label would be Pause when the clip was playing. My first attempts at accomplishing this did not go well. The use of innerHTML, for example, looked tacky. So, I created two buttons. Only one button would appear at a time through the use of display directives in the style element, as shown in Listing 4 and JavaScript, as shown in Listing 5. Notice it is necessary to position the buttons explicitly to place them under the rotated video element.

Listing 4: Style directives for the buttons

The startplay function is invoked two ways: as an event handler for the canplaythrough event and as the value for the onClick attribute in the button tag displaying Play. The pauseplay function is invoked through the setting of the onClick attribute in the button tag displaying Pause. My code extracts the values for the variables v, pb, and pzb each time because setting them once in the init function was not working. Listing 5 shows the functions and the relevant HTML markup in the body element.

Listing 5: The functions for the buttons and the events
Listing 5: The `startplay` and `pauseplay` functions along with the HTML markup for the buttons

The last change I made was to use the rounded border feature as an additional clause in the style directive for `#vid`: `border: medium; border-radius: 25px;`

Enjoy your video clips that originated as MOV (or anything else)!

**Note:** These applications have been tested by me using Firefox, Chrome, Opera and Safari on a PC. Friends have reported some success on an iPhone, an iPad and on two Android phones. The iPhone and one of the Androids did not have rounded corners and required extra clicks for play. Firefox appears to be the only browser recognizing the `canplaythrough` event. Safari on a PC did not support the rounded border corners but the iPad did. All browsers on PC recognize the `ended` event and so loop the video.

**Learn more**

There are many sources, online and in-print and some sort of e-books, for learning HTML5 and JavaScript techniques. Here are links to my recent books and website for this example:

- **The Essential Guide to HTML5: Using Games to learn HTML5 and JavaScript**, [http://www.friendsofed.com/book.html?isbn=9781430233831](http://www.friendsofed.com/book.html?isbn=9781430233831). This is a text for beginners at programming as well as more experienced programmers who want to learn about HTML and JavaScript, including the new features of HTML5, such as video.

- **HTML5 and JavaScript Projects**, [http://www.apress.com/9781430240327](http://www.apress.com/9781430240327). This book is more advanced than the first one. It contains several applications involving video, including bouncing video, combining media with Google Maps for a quiz, and a jigsaw turning into a video clip.

- To see the final version of the application and to view all the source code, go to [http://faculty.purchase.edu/jeanine.meyer/html5/playbrotated2.html](http://faculty.purchase.edu/jeanine.meyer/html5/playbrotated2.html).

Jeanine Meyer lives just north of New York City and currently teaches at Purchase College/SUNY after many years at IBM, doing research on robotics and manufacturing and consulting on educational grants. She likes providing programming examples for her Mathematics/Computer Science and New Media students and really, really likes working with images and video clips of her granddaughter and other family members.