The video element is one of the most exciting new facilities provided in HTML5. In this article, I describe an application in which the viewer clicks on buttons and portions of a video clip are played. It demonstrates the use of video element events and properties.

It may happen that you have a single video clip and decide that you want select portions of the clip to be played subject to user action in your application. Of course, one option is to use video editing software to produce multiple video clips. Another option is to stick with the original clip (which means three files of the same video in the three formats recognized by the browsers) and use features of the HTML5 video element to start and stop play at selected points. My demonstration example features a little story with buttons that will invoke different sections of the single video clip. Figure 1 shows the opening screen of my demonstration example.

If the viewer/user clicks on any of the buttons, an appropriate part of the video will play. For example, this is the end of the video clip that starts when clicking on the has a tumble herself button.

At anytime, users can use the built-in controls to play the whole video from start to finish or to hit pause or play any number of times. The format of the controls vary among the browsers, but they all provide the same functions. There are pause and play icons and a progress bar indicating the time position within the video clip. A slider providing audio volume control and an icon for full screen are available. Figure 3 shows a screen capture somewhere in the middle of the portion of the video played by the races to deliver the book button. The Chrome, Safari, Opera and Firefox browsers only show the controls when the video is paused.

The buttons can be clicked in any order. Moreover, you can click a button when the video is playing to change which portion of the video is being played or re-start any portion.

**Brief note on video**

This article is not intended to be a complete introduction to the use of the HTML5 video element. Many sources, including two of my books cited in the Learn More section, are available and my first article in jsmag (April 2012) described using video and other graphics to make a collage. The next article (May 2012) discussed handling the orientation problem that may arise from QuickTime movies. Here I just remind you that the first step, after acquiring your video, is to use a video format converter such as Miro Video Converter to produce video clips in the multiple formats (webm, mp4, and ogg). That is, for each video, you prepare 3 video files. Of course, if you know what browser your audience will be using OR if the browsers ever all settle on one format, you only need produce one file.

The next step is to code the video element for your video. Listing 1 shows the video element for this application. Notice that I have specified the controls attribute to provide the standard controls for the video element. You may decide to NOT do this for your particular application if you do not want to let your user directly...
control the video independent of your programming. The video element references the three files I created from the QuickTime .mov file sent to me featuring Annika (my granddaughter).

If I did nothing else, the video would appear when the files are loaded, most likely slightly after the text of the story appeared. However, I wrote a style directive that made the video initially invisible. I explain this in the 'Putting it all together' section.

**Video element currentTime property**

The HTML5 video element has properties that can be referenced or modified. One such property is the currentTime. This indicates the position in seconds of the video playback. I repeat: the unit is seconds, unlike the unit for the end point.

More accurately, it appears to fire frequently enough, that is, with a fine enough resolution, to support this application. My code sets up an event handler for the event and checks if the end point has been reached. The code is

```javascript
v.addEventListener("timeupdate",check,false);
```

Here check is the name of my function for handling the event and false indicates that I do not want this event to bubble up for any other application.

An alternative approach is to use the JavaScript setTimeout function.

This may provide finer level of control at the cost of some calculations to calculate the elapsed time as opposed to the end point of the video playback. I urge you to look over the video properties and video events so that you have some sense of what is available to you. In particular, if you make use of long videos, you may want to make use of specific events and properties to check that the video has been loaded successfully before starting your application.

**The timeupdate video event or an alternative approach**

There are (at least) two approaches to determining when a video has reached an end point. The video element has an event called timeupdate that fires when the playback position has changed. The timeupdate event or an alternative approach.

The timeupdate video event will occur many, many times, the exact frequency dependent on the browser. The W3C specification implies that it must be at least at the resolution of frames, typically 24 times/second. The check function compares the currentTime with the value of this function. If currentTime is greater than or equal to this function, the code causes the video to pause and removes the listening for the event. If it is not true, the check function simply ends.

```javascript
Listing 1: The body element of the application

Looking back to Listing 1, notice that the start function is called by the action of each of the button elements, and each time with two parameters, indicating the start, st, and the endpoint, ft, of the section of the video.

The definition of the init, start and check functions are shown in Listing 2.

The init function, invoked in the standard way through the onload attribute setting of the body element, positions the video element based on calculations using properties of the window.

The start function stores the endpoint time in a global variable named this function. It resets the currentTime to the value of st. The start function also sets up the video timeupdate event that will check for the finish. The event handler is the function check. The start function makes the video visible (display="block") and starts play.

The timeupdate video event will occur many, many times, the exact frequency dependent on the browser. The W3C specification implies that it must be at least at the resolution of frames, typically 24 times/second. The check function compares the currentTime with the value of this function. If currentTime is greater than or equal to this function, the code causes the video to pause and removes the listening for the event. If it is not true, the check function simply ends.

```javascript
Listing 2: The body element of the application

Putting it all together

I use the style section to make the video initially invisible. The CSS directive is

```css
video {position:absolute; z-index:10; display:none;}
```

The application makes use of three functions and a few global variables. Listing 1 shows the whole body element of the application.
I did consider making the video be invisible. This would be inserting the statement

```javascript
v.style.display="none";
```

in the check function after the `v.pause();` statement. I decided that I liked the freeze-frame of Annika. You need to decide what you want for your application.

**Variations**

More elaborate applications could make use of multiple videos. You could modify the `start` function to accept 3 arguments with the additional argument specifying which video.

A data-driven approach using an external file similar to the collage application described in the April issue of *JSMag* would be appropriate if more than one video clip was used. The external file would hold the base name of the video files and the program would create the video elements dynamically.

Any element, including video elements, can be positioned anywhere on the screen. This requires a `position: absolute` setting in style directives and code to specify the desired `top` and `left` properties. If video elements can be anywhere on the screen, then this opens up the idea of playing two or more videos at a time.

The code to play a section of a video can be incorporated in other ways than mouse clicks. For example, you can build a game or lesson application that produces different sections of a video clip depending on player or student performance. You can build a quite intricate game using parts of video clips. Using parts in place of individual clips may be easier to manage.

---

**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/booktrip1.html](http://faculty.purchase.edu/jeanine.meyer/html5/booktrip1.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 colleagues and really, really likes working with images and video clips of her granddaughter and other family members.