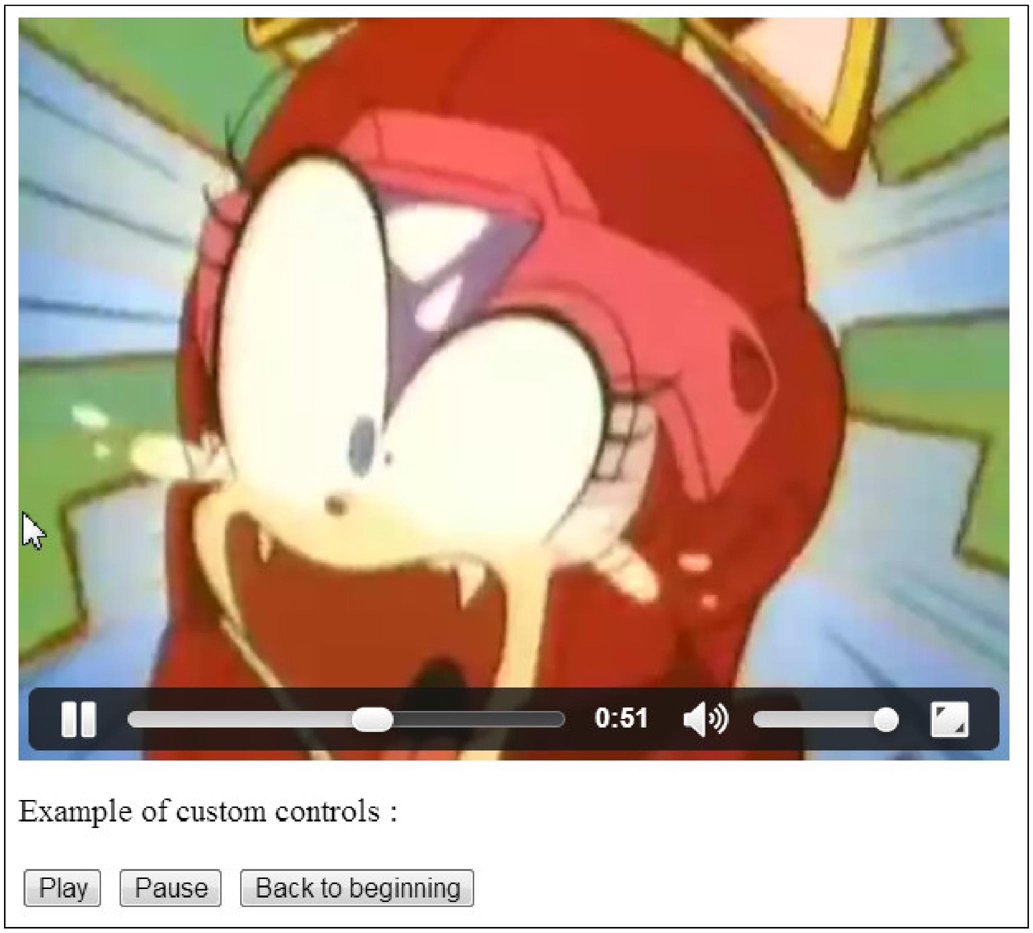
Examples of the use of the video element JavaScript API

The JavaScript API is useful for implementing playlists, making custom user interfaces and many other interesting things. The "enhanced HTML5 multimedia players" presented further on the course rely heavily on this API.

EXAMPLE 1 - HOW TO USE EXTERNAL BUTTONS TO CONTROL THE PLAYER'S BEHAVIOR

This example shows the first steps towards writing a custom video player. It shows basic usage of the JavaScript API for adding custom buttons to play/pause the video or to go back to the beginning by setting the `currentTime` property to zero.

Try it online, and look at the source code.



Source code extract:

```
<video id="vid" controls>
```

```
      <sourcesrc=http://html5doctor.com/demos/video-canvas-magic/video.webm
              type=video/webm>
    ...
    </video>
    <p>Example of custom controls:</p>
 7. <button onclick="playVideo();"style="cursor: pointer;">Play</button>

    <button onclick="pauseVideo();"style="cursor: pointer;">Pause</button>

    <button onclick="rewindVideo();"style="cursor: pointer;">
           Back to beginning</button>
     <script>
        vid = document.querySelector("#vid");
        function playVideo() {
17.        vid.play();
        }
        function pauseVideo() {
           vid.pause();
        }
        function rewindVideo() {
           vid.currentTime = 0;
        }
    </script>
```

- *Lines 7, 9 and 11*: we add a click listener to each button, in order to call a JavaScript function when the button is clicked.
- *Line 14*: using the DOM API we get the JavaScript object that corresponds to the video element we inserted in the HTML document. This line is outside a function, it will be executed when the page loads.
- *Lines 17 and 20*: we call methods from the API for playing/pausing the video.
- *Line 24*: we modify the `currentTime` property in order to rewind the video. Note that `vid.load()` also rewinds the video, shows the poster image again, but also pauses the video. By using `currentTime=0` the playback does not stop.

EXAMPLE 2 - HOW TO DETECT THE END OF A VIDEO AND START ANOTHER ONE

This example listens for the `ended` event, and calls a callback function when the video is ended.

```
<video src="video.ogv" id="myVideo">
   video not supported
</video>
<script type='text/javascript'>
  var vid = document.querySelector('#myVideo');
  vid.addEventListener('ended', playNextVideo, false);
  function playNextVideo(e) {
    // Whatever you want to do after the event, change the src attribute
    // of the video element, for example, in order to play another video
  }
</script>
```

EXAMPLE 3, APPLICATION OF THE PREVIOUS ONE - HOW TO MANAGE PLAYLISTS

This example detects the end of a video then loads the next video, changes the `src` attribute of the video element and plays the video (see the online example).

To try this example: use the progress cursor to go near the end of the first video that is being played and see how it continues with the next video.

```
<!doctype html>
<html lang="en">
<head>
 <title>Sequential Movies</title>
 <script>
   var myVideo;
   var currentVideo = 0;
   var sources = [
         "http://html5doctor.com/demos/video-canvas-magic/video.mp4",
 "http://www.archive.org/download/AnimatedMechanicalArtPiecesAtMit/P1120973_512kb.mp4"
   ];
   // Set the src of the video to the next URL in the playlist
   // If at the end we start again from beginning (the modulo
   // source.length does that)
   function loadNextVideo() {
      myVideo.src = sources[currentVideo% sources.length]
      myVideo.load();
      currentVideo++;
   }
   // callback that loads and plays the next video
  function loadAndplayNextVideo() {
      console.log("playing " +sources[currentVideo % sources.length])
      loadNextVideo();
      myVideo.play();
  }
  // Called when the page is loaded
  function init(){
      // get the video element using the DOM api
      myVideo =document.querySelector("#myVideo");
      // Define a callback function called each time a video ends
      myVideo.addEventListener('ended',loadAndplayNextVideo, false);
      // Load the first video when the page is loaded.
      loadNextVideo();
  }
```

```
</script>
</head>
<body onload="init()">
    <video id="myVideo" controls></video>
</body>
</html>
```

- *Line 8*: the JavaScript array that contains the URLs of the videos in the playlist. In this example, we've got only two of them, but if the array is larger the example will still work.
- *Line 42*: When the page is loaded, an `init()`function is called.
- *Lines 32 - 38*: we used the DOM to get the JavaScript object corresponding to the video element, then define a listener for the `ended`event. Each time a video will end, the`loadAndPlayNextVideo()` callback will be called. As the video element has no src attribute by default, we also preload the first video (call to`loadNextVideo()` at line 38).
- *Lines 16 - 20*: the `loadNextVideo()` function uses a variable called `currentVideo` that corresponds to the index of the current video. By setting`myVideo.src = sources [currentVideo % sources.length]`, we set the src of the video element to sources[0], then to sources[1], and as we increment the `currentVideo` index each time (line 19), if it becomes greater than 1, the modulo (the "%" symbol is the modulo in JavaScript) will make it "loop" between 0 and the number of videos in the playlist. In other words, when the last video ends, it starts back to the first one.