Extended examples

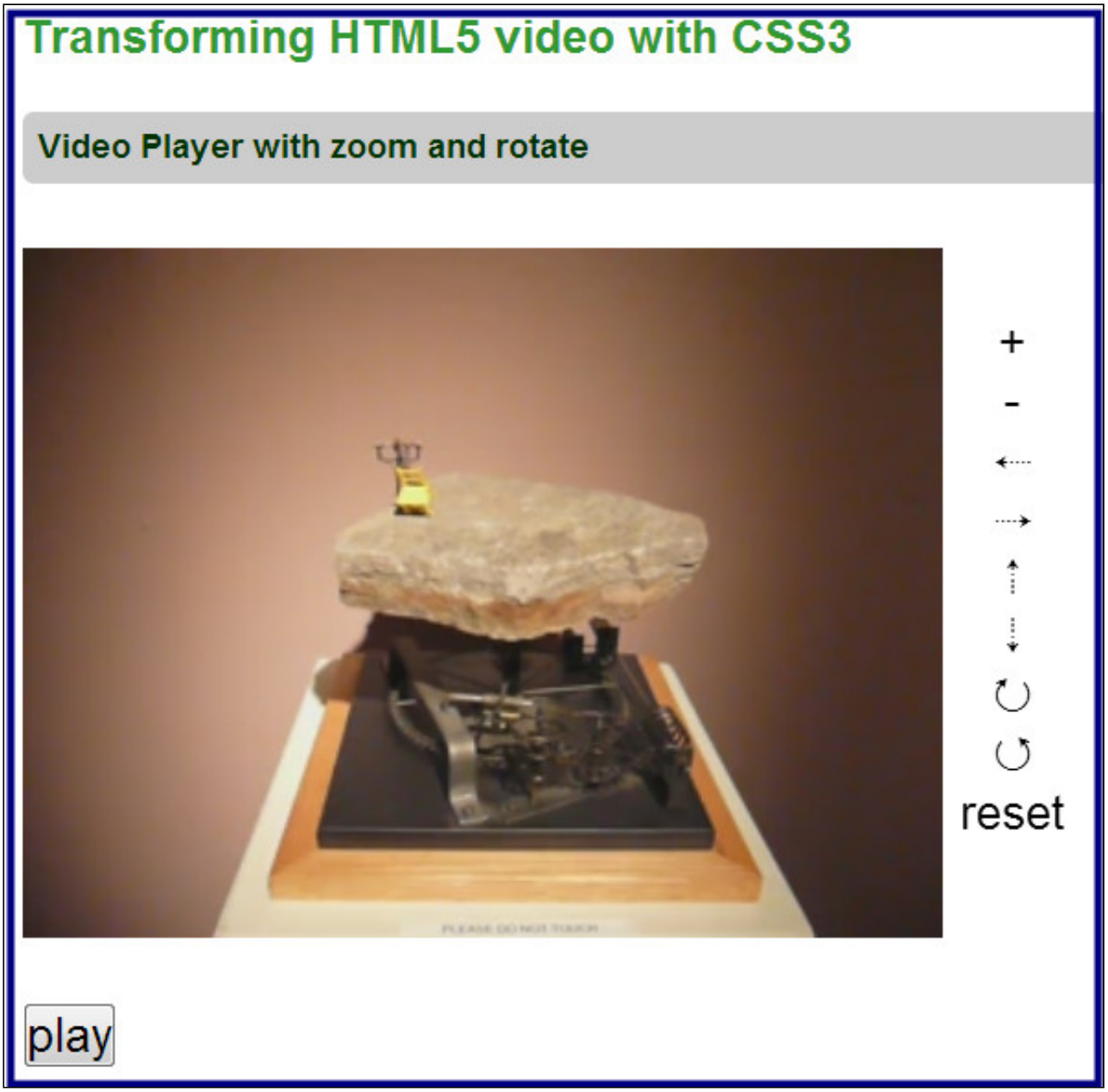
In this section, we propose six extended examples that use more JavaScript and more complex CSS manipulation. They might be a little hard to understand if you are a JavaScript beginner, but don't be afraid to try and test them, look at the code, etc.

Some examples are given "as is", such as the custom video player that uses SVG (at the end of the page); if you are interested, you may view the code.

EXAMPLE 1: A PLAYER SHOWING THE USE OF EVERY TYPE OF CSS3 TRANSFORMATION

Please see this example online, originally written by Chris Heilmann, and tuned by us ;) The editable source code is here at JS Bin.

Don't forget to click the JavaScript and CSS tabs at JS Bin in order to display the JavaScript code that creates the buttons on the right of the video, and the CSS3 that processes the different clicks and applies CSS3 transforms.



This example shows a lot:

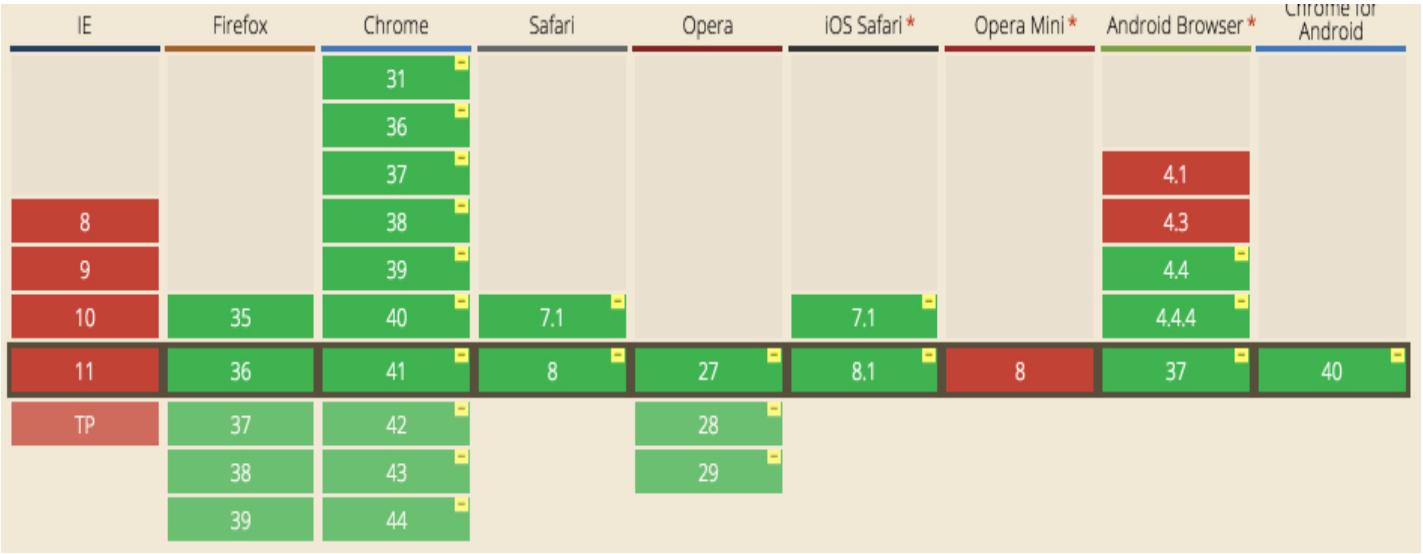
- It uses the HTML5 elements `<nav>`, `<footer>`, `<header>`.
- It shows the use of CSS3 2D transformations (scale, translate, and rotate).
- It shows how to handle DOM events using JavaScript and how to modify CSS properties of the `<video>` element from JavaScript.

EXAMPLE 2: APPLYING CSS3 FILTERS TO A VIDEO IN REAL TIME

Please see this example online. Play the video and then click on the video while it's playing. This will change in real-time the CSS class of the video element. Each class uses the `filter` property with different values.

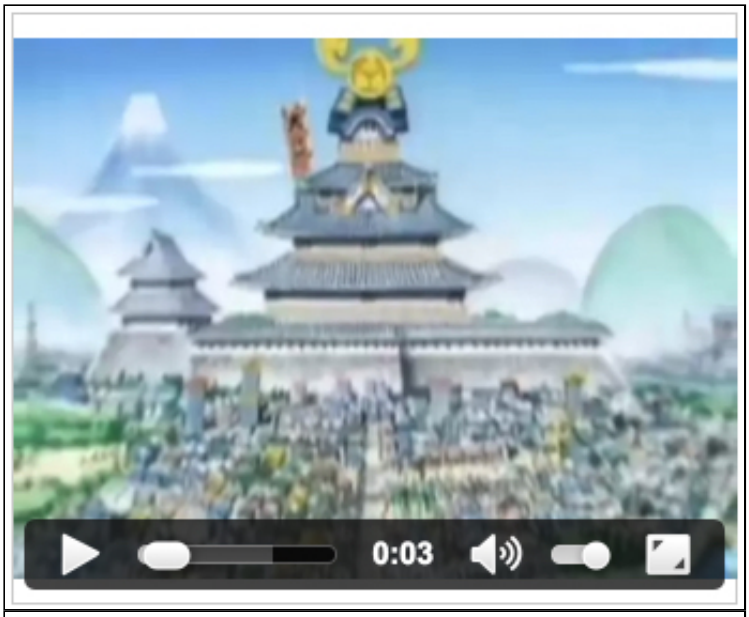
Note that CSS filters are not yet 100% supported by the major browsers. You still need to use prefixed versions of the CSS properties, as shown below (this table is taken fromcaniuse.com).

CSS filter support (green squares with a small yellow part in the top right corner) means that a prefix is needed, like `-webkit-filter`, or `-moz-filter` or `-o-filter`):



There is an up-to-date version of this table.

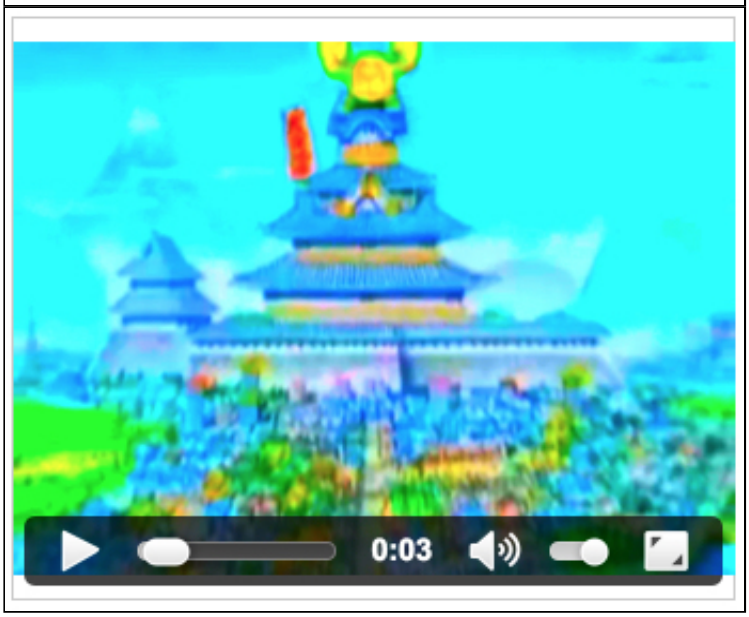
Below, you will find images obtained with different filter values:


0:03


0:03


0:03


0:03


0:03

Use `<video class="blur">` for example, to obtain a blurry video. This complete example changes the CSS class associated to the video element, on the fly in a `mouseclick` listener callback.

Here, we define the CSS classes used in the example:

```
.blur {
    filter: blur(3px);
}
.brightness {
    filter: brightness(5);
}
.contrast {
    filter: contrast(8);
}
.hue-rotate {
    filter: hue-rotate(90deg);
}
.hue-rotate2 {
    filter: hue-rotate(180deg);
}
.hue-rotate3 {
    filter: hue-rotate(270deg);
}
.saturate {
    filter: saturate(10);
}
.grayscale {
    filter: grayscale(1);
}
.sepia {
    filter: sepia(1);
}
.invert {
    filter: invert(1)
}
```

This extract from the source code explains how to set a `mouseclick` listener and how to change the value of a CSS class attribute on the fly:

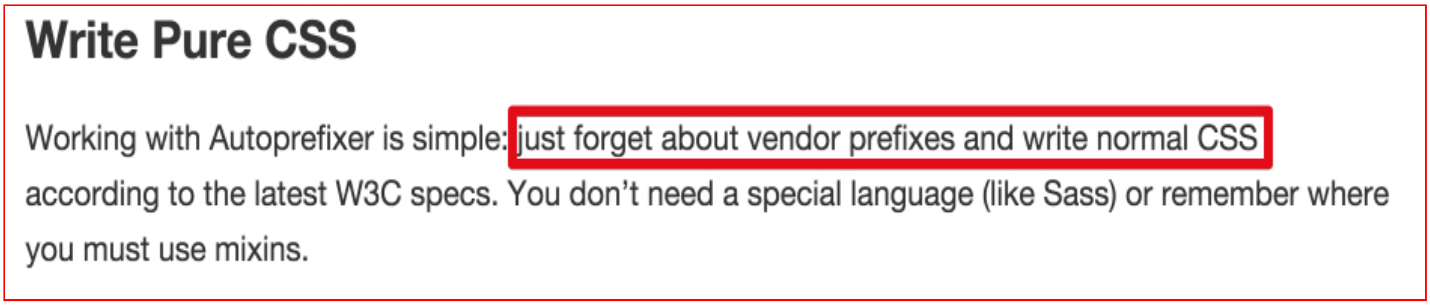
```
<video id="output" controls autoplay>
   <sourcesrc=http://html5doctor.com/demos/video-canvas-
magic/video.webm
           type=video/webm>
   <sourcesrc=http://html5doctor.com/demos/video-canvas-
magic/video.ogg
           type=video/ogg>
   <sourcesrc=http://html5doctor.com/demos/video-canvas-
magic/video.mp4
           type=video/mp4>
</video>
<script>
   var output =document.getElementById('output');
   var idx = 0;
   var filters = [
     'grayscale',
     'sepia',
     'blur',
     'brightness',
     'contrast',
     'hue-rotate', 'hue-rotate2', 'hue-rotate3',
     'saturate',
     'invert',
 ''];
 function changeFilter(e) {
     var el = e.target;
     var effect = filters[idx++ %filters.length];

     if (effect) {
         el.classname = effect;
     }
     // Do not propagate the event, prevent default behavior.
     // By default, a click on a video element
pauses/unpauses the video
     // By stopping the propagation and canceling the default
behavior,
```

```
        // we stop the pause/unpause behavior when the video is
    clicked.
        // Now a click just changes the CSS filter we apply on
    the video.
        e.stopPropagation();
        e.preventDefault();
     }
    output.addEventListener('click',changeFilter, false);
    </script>
    <style>
    #output {
        width: 307px;
        height: 250px;
        background: rgba(255,255,255,0.5);
        border: 1px solid #ccc;
    }
    .blur {
        filter: blur(3px);
52. }
    .brightness {
        filter: brightness(5);
    }
    ...
    </style>
```

Notes about "prefixed CSS properties" and a good tool that will add prefixes for you!

If you look at the previous CSS file, the CSS filter property is not prefixed (for example, `-webkit-filter` instead of `filter`), whereas some browsers still require prefixes.

autoprefixer.js is a very practical tool. It's a simple library you can include in your HTML page that will add prefixes on the fly when needed. It uses the caniuse.com tables and also relies on the W3C specifications, so we recommend it!



Typical use: just add some lines of code to the `<head>` element of your HTML page.

```
<head>
    <scriptsrc="https://rawgithub.com/ai/autoprefixer-
rails/master/vendor/autoprefixer.js"></script>
<style type="unprocessed"id="AutoprefixerIn">%css%</style>
<style id="AutoprefixerOut"></style>
<script>
    AutoprefixerSettings = ""; // Specify here the browsers
you
                               // want to target or leave
empty
    document.getElementById("AutoprefixerOut").innerHTML=
              autoprefixer(AutoprefixerSettings|| null)

.process(document.getElementById("AutoprefixerIn")
                        .innerHTML)
              .css;
</script>
</head>
```

EXAMPLE 3: HOW TO TRACK ALL POSSIBLE EVENTS AND MANIPULATE MANY PROPERTIES

This example also shows how to handle failures. See the code and play with this example online.

timeupdate 12.353
timeupdate 12.098
timeupdate 11.842
timeupdate 11.598
timeupdate 11.345
timeupdate 11.100
timeupdate 10.847
timeupdate 10.593
timeupdate 10.347
timeupdate 10.094
timeupdate 9.841
seeked

Here is an example of a piece of code for handling errors during video playback:

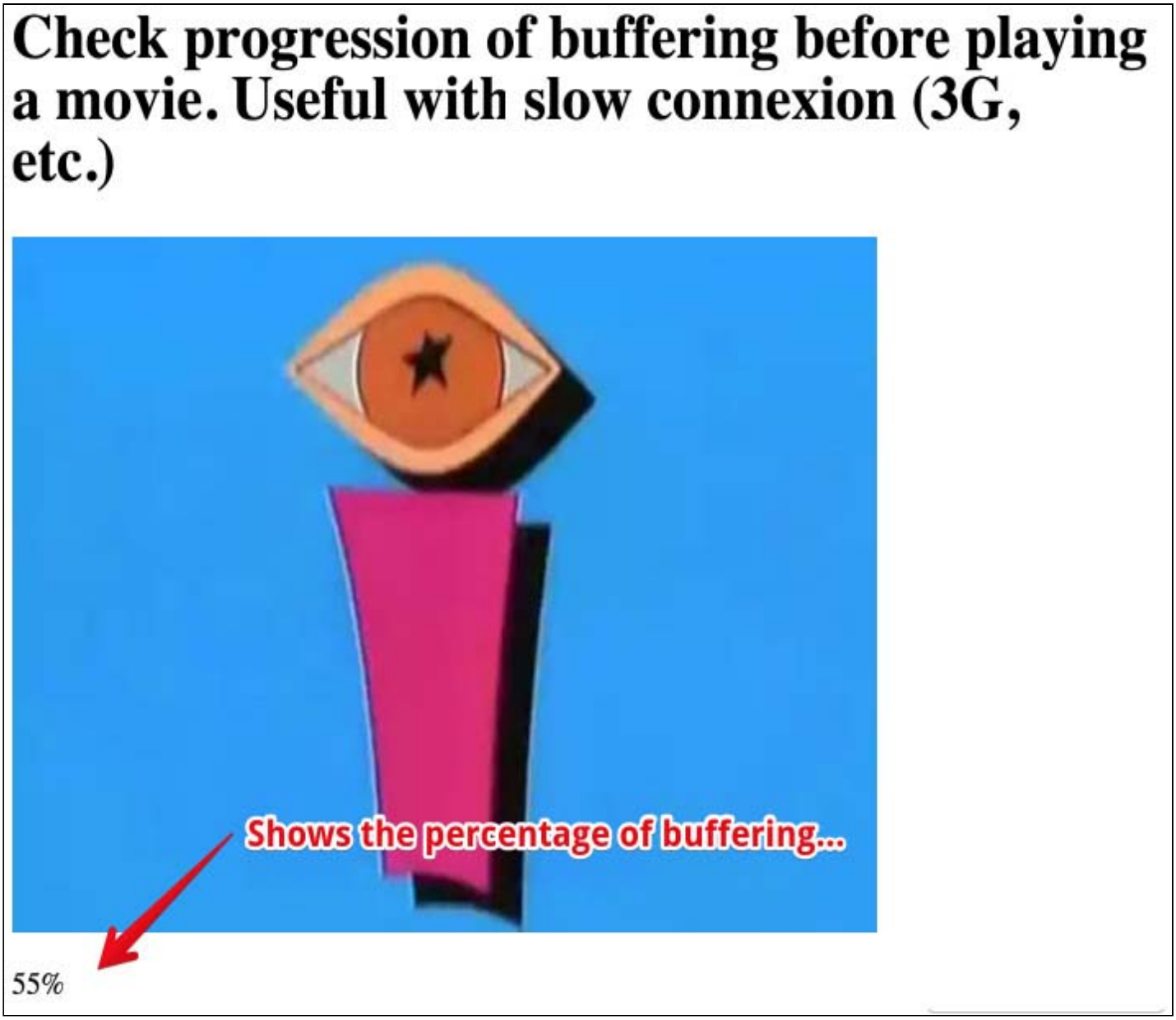
```
. . .
```

```
    vid.addEventListener('error',function(evt) {
        logEvent(evt,'red');
    }, false);
    ...
    function logEvent(evt, color) {
        switch (evt.type) {
            ...
            case 'error':
                var error =document.querySelector('video').error;
                switch (error.code) {
                  caseerror.MEDIA_ERR_ABORTED:
                     note.innerHTML ="fetching aborted at the
user's request";
                     break;
                  caseerror.MEDIA_ERR_NETWORK:
                     note.innerHTML = "a network error caused the
browser to stop fetching the media";
                     break;
                  case error.MEDIA_ERR_DECODE:
                     note.innerHTML = "an error occurred while
decoding the media";
                     break;
                  caseerror.MEDIA_ERR_SRC_NOT_SUPPORTED:
                     note.innerHTML = "the media indicated by the
src
                                        attribute was not
suitable";
                     break;
                  default:
                     note.innerHTML = "an error occurred";
                     break;
                }
                break;
            }
     ...
    }
```

EXAMPLE 4: HOW TO DISPLAY A PERCENTAGE OF BUFFERING

WHEN USING A SLOW CONNECTION

See the example online here too.



Note that on mobile phones, the video does not start until the user presses the play control or clicks on the video picture. Using the "`canplaythrough`" event is a trick to call a function that starts the video player as soon as the page is loaded on desktop. This event is not supported by mobile devices, so if you try this example on a mobile, the video will not start automatically.

As the Apple Developer Web site explains it: "The `buffered`property is

a `TimeRanges` object: an array of start and stop times, not a single value. Consider what happens if the person watching the media uses the time scrubber to jump forward to a point in the movie that hasn't loaded yet—the movie stops loading and jumps forward to the new point in time, then starts buffering again from there. So the `buffered` property can contain an array of discontinuous ranges. The example simply seeks the end of the array and reads the last value, so it actually shows the percentage into the movie duration for which there is data. "

```
<!doctype html>
<html lang="en">
  <head>
    <title>JavaScript Progress Monitor</title>
    <script>
      function getPercentProg() {
          var myVideo =document.getElementsByTagName('video')
[0];
          var endBuf =myVideo.buffered.end(0);
10.
  var soFar = parseInt(((endBuf /myVideo.duration) * 100));

document.getElementById("loadStatus").innerHTML= soFar + '%';
      }
      // Will be called as soon as the page is ready on
desktop computer,
      // Only when a user clicks on play control or image on
mobile
      function myAutoPlay() {
          var myVideo =document.getElementsByTagName('video')
[0];
          myVideo.play();
19.       }
20.
21.       function addMyListeners(){
          var myVideo =document.getElementsByTagName('video')
[0];

myVideo.addEventListener('progress',getPercentProg, false);
```

```
            // Calls autoplay only if the device is adapted

    myVideo.addEventListener('canplaythrough',myAutoPlay, false);
        }
  </script>
</head>
<body onload="addMyListeners()">
    <h1>Check progression of buffering before playing a
movie. Useful withy
        slow connexion (3G, etc.)</h1>
    <div>
      <video controls>
          <sourcesrc=http://html5doctor.com/demos/video-
canvas-magic/video.webm
                  type=video/webm>
          <sourcesrc=http://html5doctor.com/demos/video-
canvas-magic/video.ogg
                  type=video/ogg>
          <sourcesrc=http://html5doctor.com/demos/video-
canvas-magic/video.mp4
                  type=video/mp4>
      </video>
      <p id="loadStatus">Buffering...</p>
    </div>
</body>
</html>
```

EXAMPLE 5: HOW TO USE SVG ELEMENTS AS EXTERNAL CONTROLLERS

This is the ultimate way of doing a real custom player: redesign your own controls using SVG shapes! This example is given "as is" for those of you who may be curious. An SVG course from W3C might be available on W3Cx one of these days. Stay tuned ;)

Try it online!

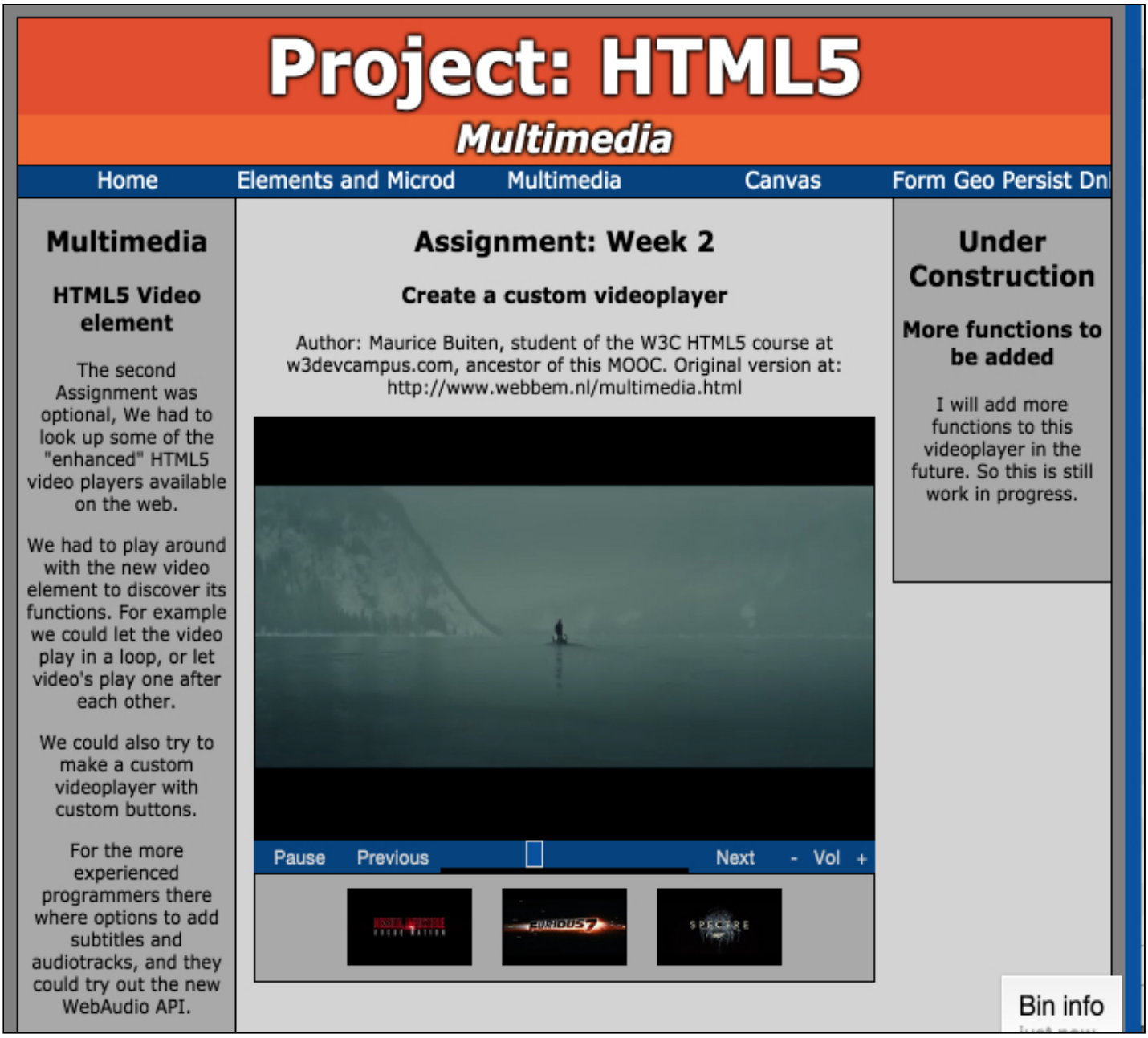
EXAMPLE 6: A CUSTOM VIDEO PLAYER WRITTEN BY A STUDENT WHO TOOK A PRECURSOR VERSION OF THIS MOOC

This is more an example than a tutorial. Maurice, a student who followed the precursor version of this MOOC at the w3devcampus.com Web site, had the assignment to write a custom video player with playlist, video thumbnails, custom play/pause/next/previous/volume controls, and present it in a Web page that used a nice layout based on the new structuring elements seen during Week 1.

Here is the online example on JS Bin, by Maurice Buiten, and here is the original version.

We recommend looking at the source code; you will learn many things related to the Week 1 course.

Screenshot:


Project: HTML5
Multimedia
Home
Elements and Microd
Multimedia
Canvas
Form Geo Persist Dn
Multimedia
HTML5 Video element
The second Assignment was optional, We had to look up some of the "enhanced" HTML5 video players available on the web.
We had to play around with the new video element to discover its functions. For example we could let the video play in a loop, or let video's play one after each other.
We could also try to make a custom videoplayer with custom buttons.
For the more experienced programmers there where options to add subtitles and audiotracks, and they could try out the new WebAudio API.
Assignment: Week 2
Create a custom videoplayer
Author: Maurice Buiten, student of the W3C HTML5 course at w3devcampus.com, ancestor of this MOOC. Original version at: http://www.webbem.nl/multimedia.html
Pause
Previous
Next
- Vol +
Under Construction
More functions to be added
I will add more functions to this videoplayer in the future. So this is still work in progress.
Bin info