# Styling media players with CSS3

The `<video>` and `<audio>` elements are just like other HTML elements, so CSS3 can be used for styling, including CSS3 transitions, animations and transforms. This was not possible with Flash technology.

## AN EXAMPLE OF AN AUDIO PLAYER WITH SOME STYLE

### Add some decoration and styling

You can try this example online at JS Bin.

To add some styling to the basic example we saw when we introduced the `<audio>` element, we just add a `<figure>` with two children: an `<img>` and a `<figcaption>`. Inside the`<figcaption>` we add the `<audio>` element from the previous example.

MOVE THE MOUSE POINTER OVER THIS PLAYER'S ELEMENTS!

HTML source code:

```
<figure id="figaudio1">
  <img id="imghorse" width="200"

 src="http://upload.wikimedia.org/wikipedia/commons/d/d4/Nokota_Horses.jpg"
     alt = "a horse"/>
  <figcaption id="figcptionaudio1"> Press Play to hear the horse!
    <audio controls="controls">

 <sourcesrc="https://dl.dropboxusercontent.com/u/1631516/horse.ogg"
               type="audio/ogg" />

 <sourcesrc="https://dl.dropboxusercontent.com/u/1631516/horse.mp3"
               type="audio/mp3" />
       Your browser does not support the audio element.
       Download the audio/video in

 <ahref=”https://dl.dropboxusercontent.com/u/1631516/horse.ogg”>OGG</a>

 or <ahref=”https://dl.dropboxusercontent.com/u/1631516/horse.mp3”>MP3</a>
        format.
   </audio>
 </figcaption>
</figure>
```

CSS source code:

```
#figaudio1 {
    width : 420px;;
    text-align:center;
    padding : 6px;
    background : white;
    margin : 0 11px 0px 0;
    border :solid 1px #888888;
    border-radius : 8px ;
}

#figcptionaudio1 {
    font-size : .8em;
    padding : 6px 8px;
    background : #dddddd;
    display :block;
    text-align :center;
    font-family : georgia, serif;
    font-style : italic;
    border-radius : 7px ;
}

#figaudio1 > img {
    background : #eeeeee;
    padding : 5px;
    border : solid 1px #444444;
}

/* For audio and img transitions/animation */
audio, #figaudio1 > img {
    transition:all 0.5s;
}

#figaudio1 > img:hover {
    box-shadow: 15px 15px 20px rgba(0,0,0, 0.4);
    transform: scale(1.05);
}

audio:hover, audio:focus, audio:active {
    box-shadow: 15px 15px 20px rgba(0,0,0, 0.4);
    transform: scale(1.05);
}
```

## CHANGING THE SIZE OF A VIDEO ON THE FLY USING CSS3 TRANSFORMS

### Resizing and rotating a video as the mouse pointer comes over it

See this example online (where you can modify the code on the fly) or just play the following video, and move the mouse pointer in and out of the video while it's playing.

This example uses the pseudo CSS class `:hover` in order to track the `mouseover` event. On mouseover, it uses a CSS3`transition` property that interpolates the changes in the scale and orientation of the video element (done using a `transform`CSS property).

The corresponding HTML source code is:

```
<video id="w3devCampusVideo" autoplaycontrols>
     <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>
```

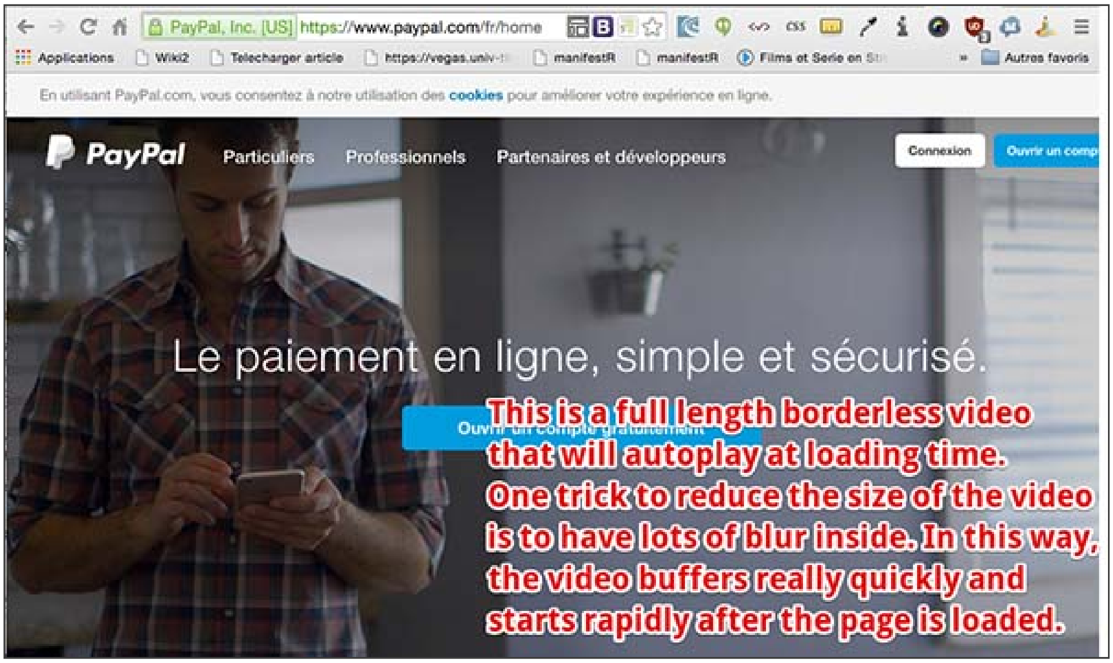
... and the CSS source code is as follows:

```
#w3devCampusVideo {
    width: 300px;
    transition: all 0.5s ease-in-out;
}

#w3devCampusVideo:hover {
    width:400px;
    transform:rotate(-5deg);
}
```

## FullScreen video that resizes and maintains ratios. Uses simple JavaScript to modify CSS properties

This is a trendy way of displaying videos (the PayPal Web site, as of June 2015, used a small size video that looped and started playing as soon as the page was loaded. One of the JsBin examples below uses this video. Note that as of October 2015 PayPal removed the video and replaced it by a still image).



En utilisant PayPal.com, vous consentez à notre utilisation des cookies pour améliorer votre expérience en ligne.

Le paiement en ligne, simple et sécurisé.

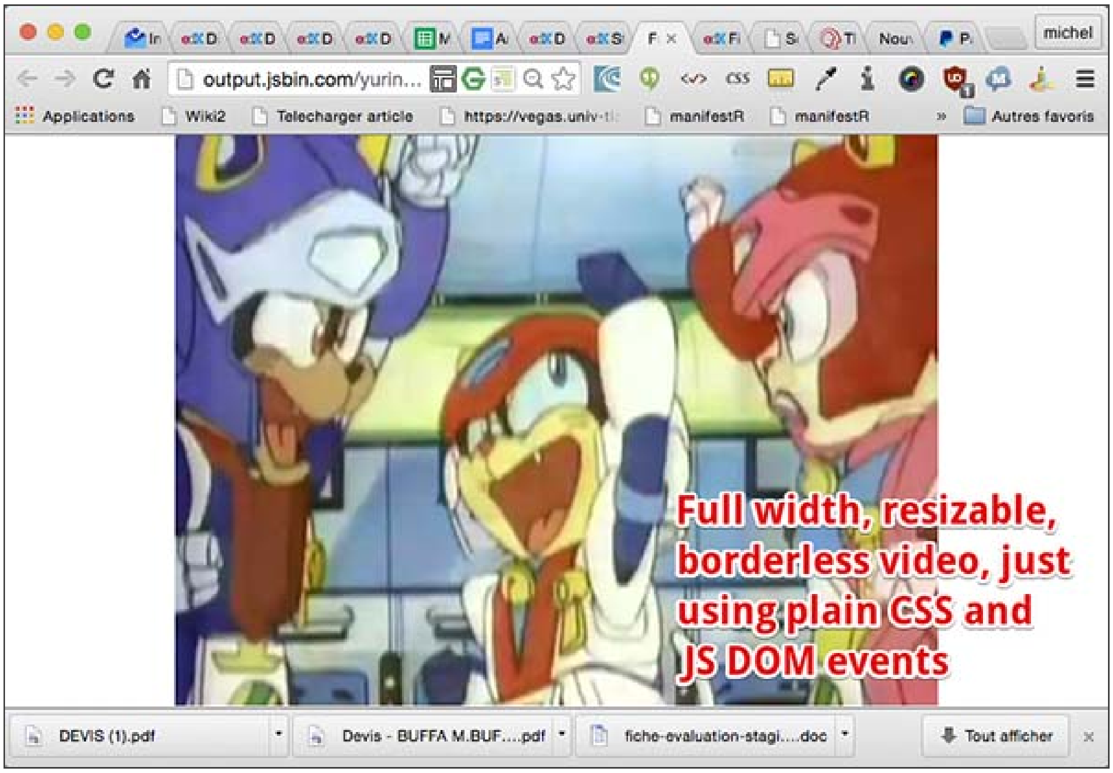
This is a full length borderless video that will autoplay at loading time. One trick to reduce the size of the video is to have lots of blur inside. In this way, the video buffers really quickly and starts rapidly after the page is loaded.

Below you will find two examples that show how to do this trick. The first is for a "regular" video, using the `<video>` and`<source>` elements. This technique can also be used on any YouTube embedded videos (see Example 2 below).

The interesting part is that we use a 100% standard (and really small and simple) JavaScript code here to handle the window`resize` events and we just set regular CSS properties `width`and `height` of the video element, to resize the video.

### Example 1: with a regular video

- Online at JS Bin



Full width, resizable, borderless video, just using plain CSS and JS DOM events

Here is the HTML code. It's really simple, just notice the `<body onload="init();">` which calls the JavaScript `init()` function right after the page is loaded.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <title>Full width video like PayPal site</title>
</head>
<body onload="init();">
    <video id="myVideo" autoplay>
      <source
          src=http://html5doctor.com/demos/video-canvas-magic/video.webm
          type=video/webm>
      <source
          src=http://html5doctor.com/demos/video-canvas-magic/video.ogg
          type=video/ogg>
      <source
          src=http://html5doctor.com/demos/video-canvas-magic/video.mp4
          type=video/mp4>
    </video>
</body>
```

Here is the CSS (remove margins, remove padding, hide parts that could overflow from the <body>):

```
body {
    margin:0;
    padding:0;
    overflow:hidden;
}
```

And now the JavaScript code:

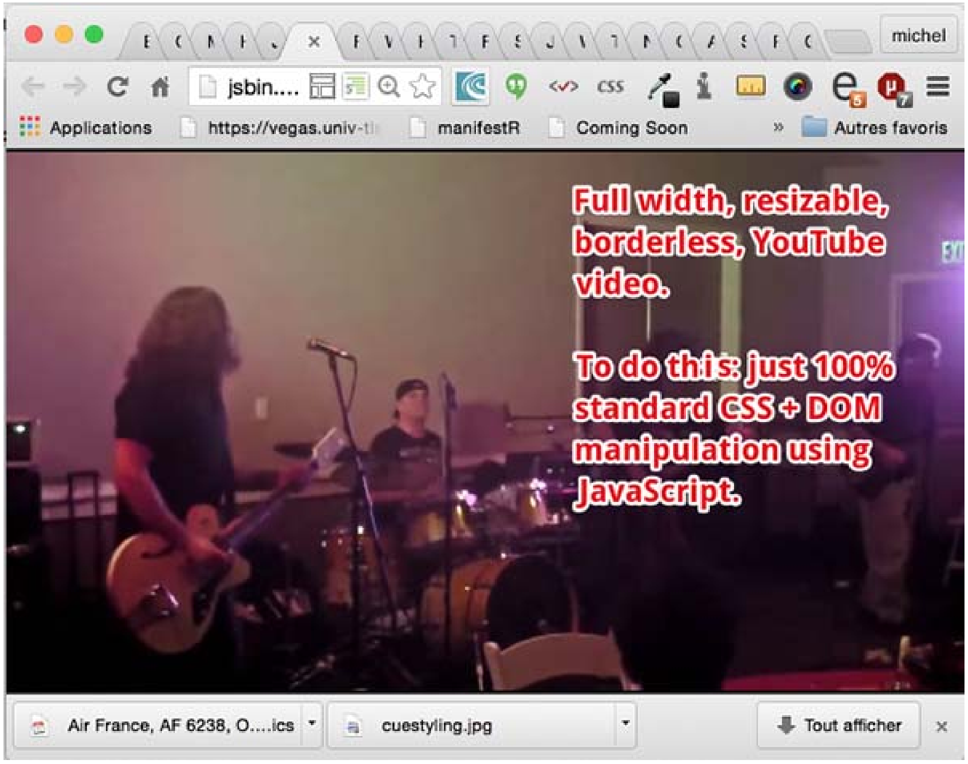
```
var video;

function init() {
   // function called when the page is loaded
   video =document.querySelector("#myVideo");
   // For initial value
   video.width = window.innerWidth;
   video.height = window.innerHeight;
   // For dealing with window resize
   window.onresize = function() {
       video.width = window.innerWidth;
       video.height = window.innerHeight;
   };
}
```

**Example 2: with a YouTube video**

- Online at JS Bin

The CSS and JavaScript codes for this example are exactly the same as in Example 1.



Full width, resizable, borderless, YouTube video.

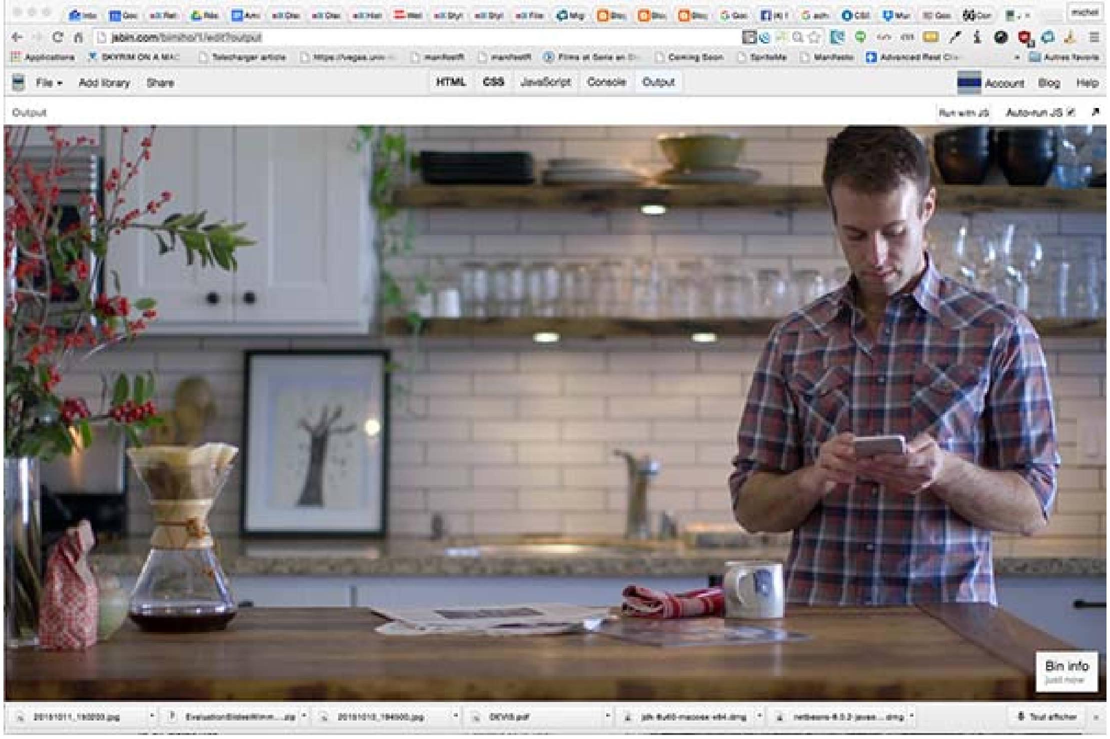
To do this: just 100% standard CSS + DOM manipulation using JavaScript.

## Full screen video, pure CSS approaches (external resources)

**Here is a JSBin with the video from the PayPal Web site, played full screen using only very simple CSS.**

In this example, the video does not rescale; it's just cropped if the browser window is resized. Enlarge your browser and you'll see a man with a phone on the right. Resize your browser and you'll see only part of the video.

Example on JsBin



CSS code:

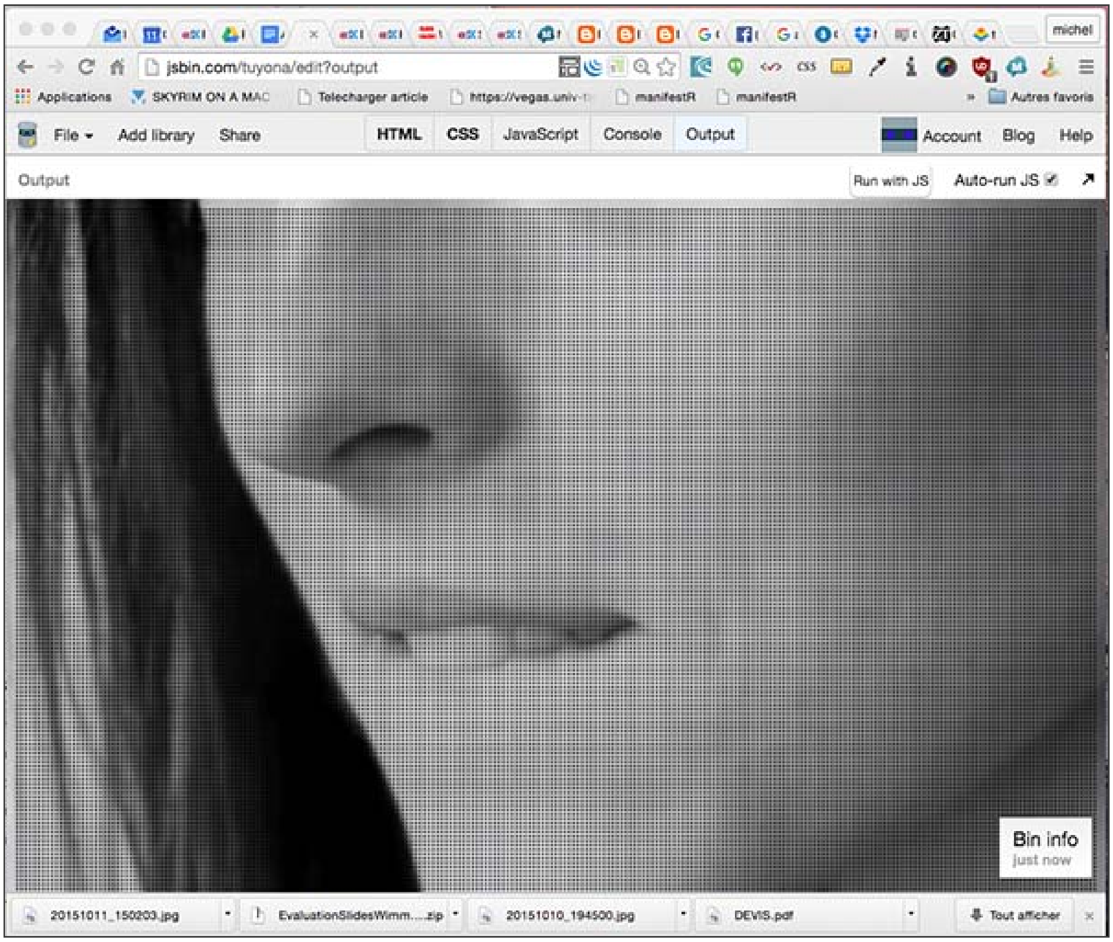
```
body {
  margin:0;
  padding:0;
  overflow:hidden;
}

video {
  width:100%;
  height:auto;
}
```

### Full screen video with CSS effects - example 2

This time the video is zoomed in so that it's much bigger than the browser's window. When we resize the browser, the part of the video that is visible adapts itself. It's not "real resize" of the video.

Try the example and read the explanation in an article by Dudley Storey. Also available as a simplified JsBin example.



HTML code:

```
<!DOCTYPE html>
<html>
<head>
   <meta charset="utf-8">
   <title>Full screen video, example from demosthene.info by </title>
</head>
<body>
<header>
 <video autoplay="" loop=""
  poster="https://s3-us-west-2.amazonaws.com/s.cdpn.io/4273/polina.jpg"
  id="bgvid">
    <sourcesrc="http://demosthenes.info/assets/videos/polina.webm"
            type="video/webm">
    <sourcesrc="http://demosthenes.info/assets/videos/polina.mp4"
            type="video/mp4">
 </video>
</header>
<section>
<h1>http://demosthenes.info/blog/777/Create-Fullscreen-HTML5-Page-
```

```
Background-Video</h1>
</section>
</body>
</html>
```

CSS code:

```
    html, body{
        color:white;
        height: 100%;
    }
    header{
        height: 100%;
        background-image:url('http://dupontcours.free.fr/IMG/dots.png'),
                         url('#');
        background-repeat: repeat, no-repeat;
10.     background-size: auto, cover;
        background-position: center center,top left;
        font-family: sans-serif;
        color: #051a00;
    }

    header video {
        position:fixed;
        top:50%;
        left:50%;
        min-width:100%;
        min-height:100%;
22.     width:auto;
        height:auto;
        z-index:-100;
        transform:translateX(-50%)translateY(-50%);
    }
```

The trick here is that:

1. the video is in the header, and the header has a plotted transparent background that is repeated in X and Y (see line 8 and 9).
2. The video is positioned so that it's origin (top left corner) is away from the visible surface (line 25), while it is set to take 100% of the surface (lines 20 and 21).

**Full screen video that resizes and keeps its ratio, using the viewport units**

**(beware, not supported by old browsers)**

Example on JsBin

This time we obtain the same result as with the first example that used JavaScript and a `resize` event. The video resizes correctly and keeps its ratio.

CSS code:

```
body {
   margin: 0;
}


video {
   position: absolute;
   width: 100vw;
   height: 100vh;
   object-fit: cover;
   object-position: center center;
}
```

## Discussion: why can't we achieve perfect resizing with only CSS and the use of properties`width=100%` and `height=100%`?

Let's use the same video to compare the different approaches again:

1. Original, using JavaScript - http://jsbin.com/yurinu/4/edit. This solution works on any browser, so we will focus on the two following methods, based on pure CSS.
2. Using CSS 100% `width` and `height` properties (no JavaScript) - http://jsbin.com/yuzavusasa/1/edit
3. Using CSS viewport units for width and height (no JavasScript) - http://jsbin.com/qamiqadazu/1/edit

Resizing the browser window shows that #1 (JavaScript) and #3 (viewport units) behave in the same way: the width or height of the video always fills the window (whichever is smaller), and we always see the whole video.

Conclusion: we can get full size video without JavaScript by using viewport units, unless we need to support some old browsers

(See http://caniuse.com/#feat=viewport-units for current support).

Setting the video to 100% `width` and `height` results in different behavior:

- 100% means 100% of the size of the `<body>` tag.
- The body tag's width is 100% of the browser window width, so the video is always full width.
- The body tag's height, however, is determined by the size of its children: the body tag's height grows and shrinks to accommodate the size of the children.
- If the browser window is made wide and short, the video is full width, the height is taller than the window, and part of the video is not visible. It seems that just using % does not get us the same effect.

## KNOWLEDGE CHECK 2.2.4 (NOT GRADED)

Using CSS, is it possible to apply geometric transformations to a video player, or to add shadows to an audio player?

○ Yes

○ No

CHECK