The Shadow DOM

INTRODUCTION

The Shadow DOM API provides DOM encapsulation: is hides what is not necessary!

This is already used by browsers' developers for `<audio>` or `<video>` elements, and also for the new `<input type=date>`, `<input type=color>` elements, etc.

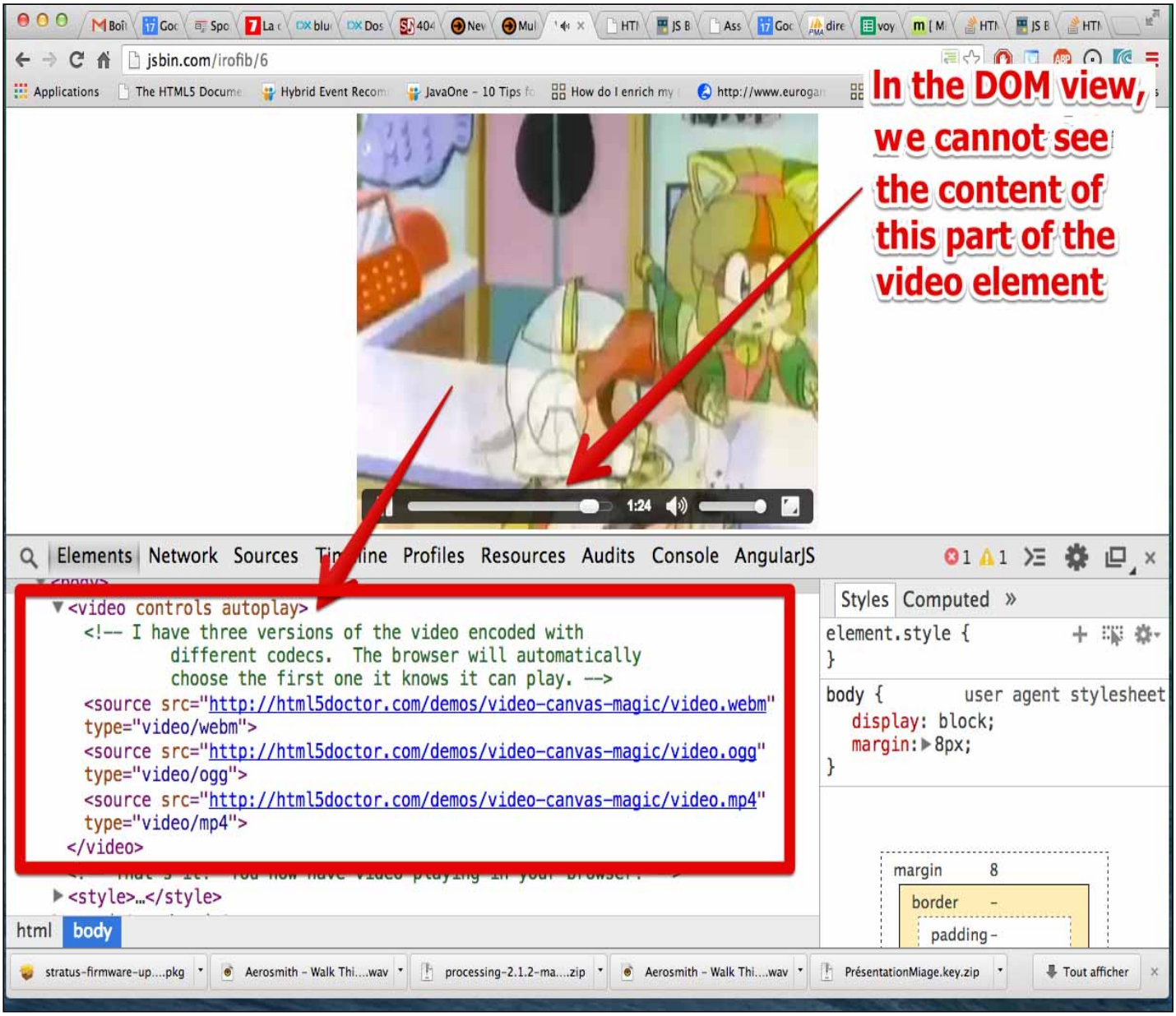
> **The three rules of Shadow DOM:**
>
> 1. With Shadow DOM, elements are associated with a new kind of node: *a shadow root*.
> 2. An element that has a shadow root associated with it is called *a shadow host*.
> 3. *The content of a shadow host isn't rendered; the content of the shadow root is rendered instead.*

AN EXAMPLE USING THE SHADOW DOM: THE`<VIDEO>` ELEMENT

Let's have a look at a simple `<video>` element in Google Chrome.

Open this JSBin example in your browser, and fire up the devtools console (F12 on Windows/Linux, Cmd-Alt-i on Mac OS):

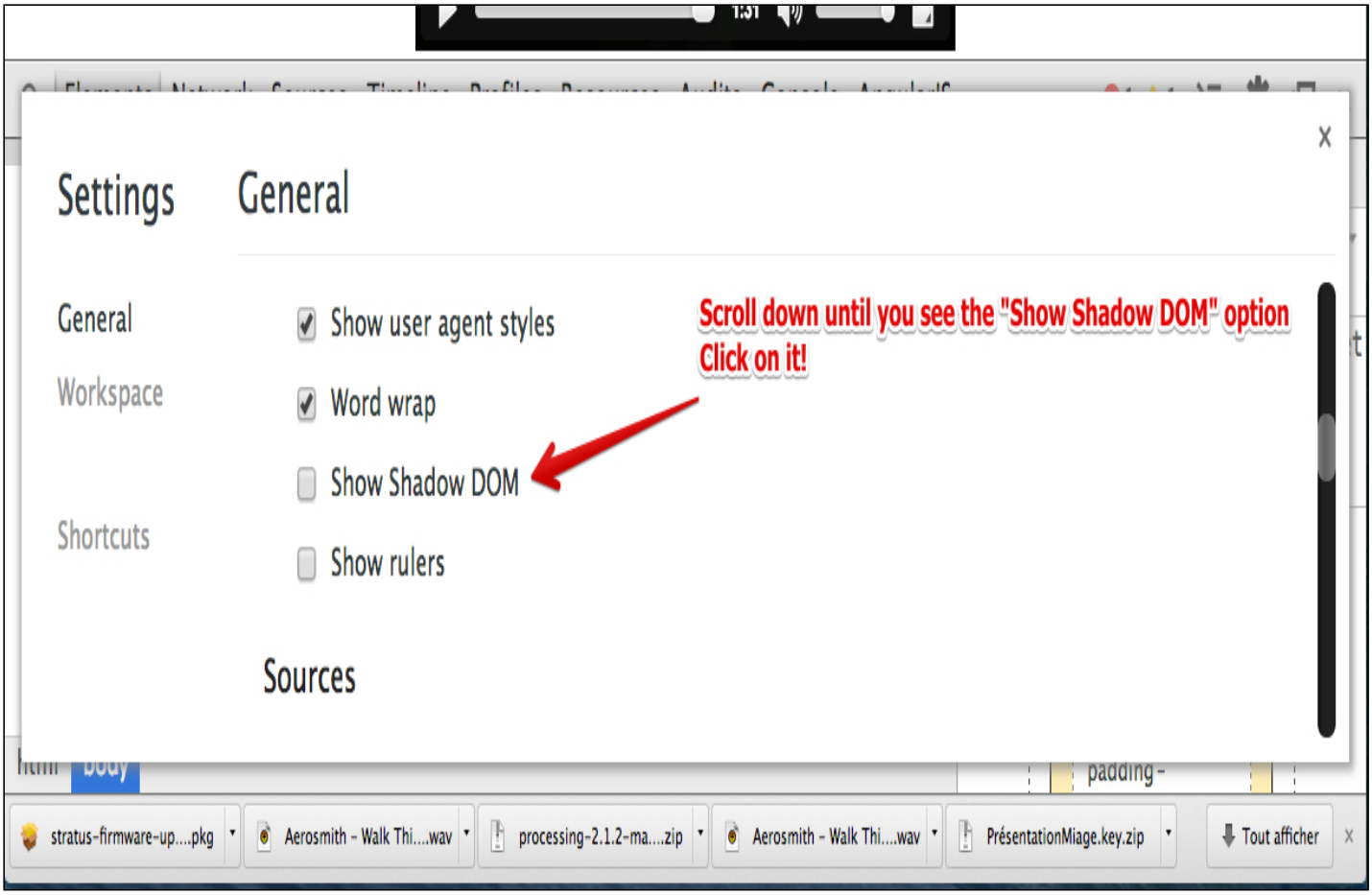
Click on the "Elements" tab in the devtools, or use the magnifying glass and click on the video, to look at the the DOM view of the video element. You will see the exact HTML code that is in this example, but you cannot see the elements that compose the control bar. You don't have access to the play button, etc.



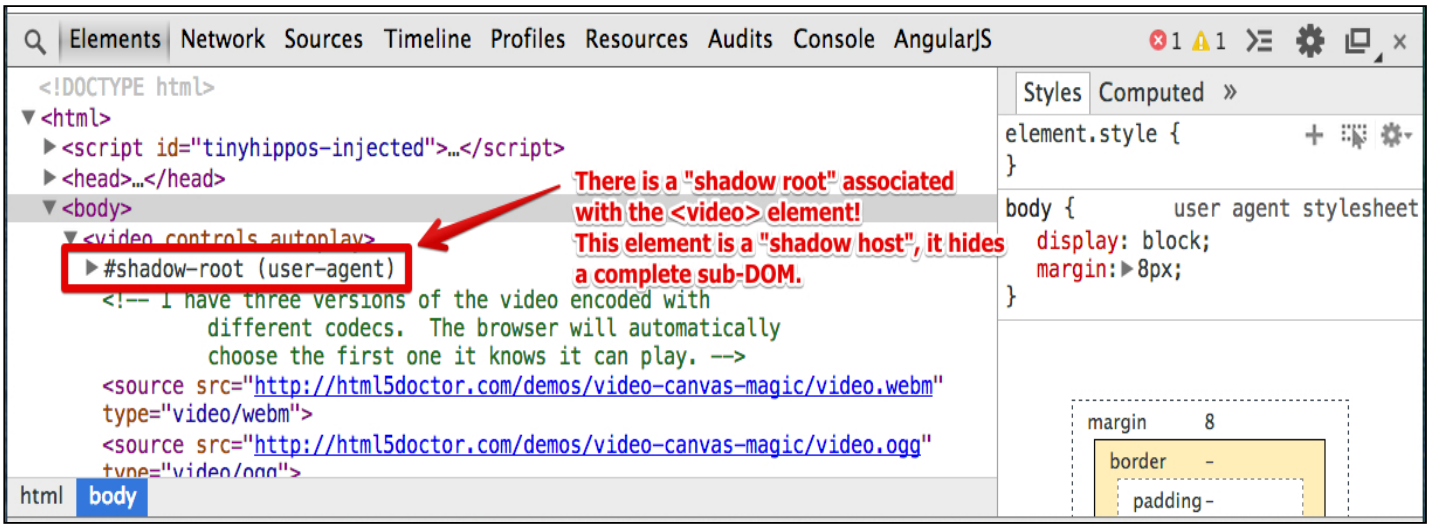
Now, let's look behind the scenes, and see the Shadow DOM associated with the `<video>` element.

First, click on the small gear icon to open the option panel of the devtools:

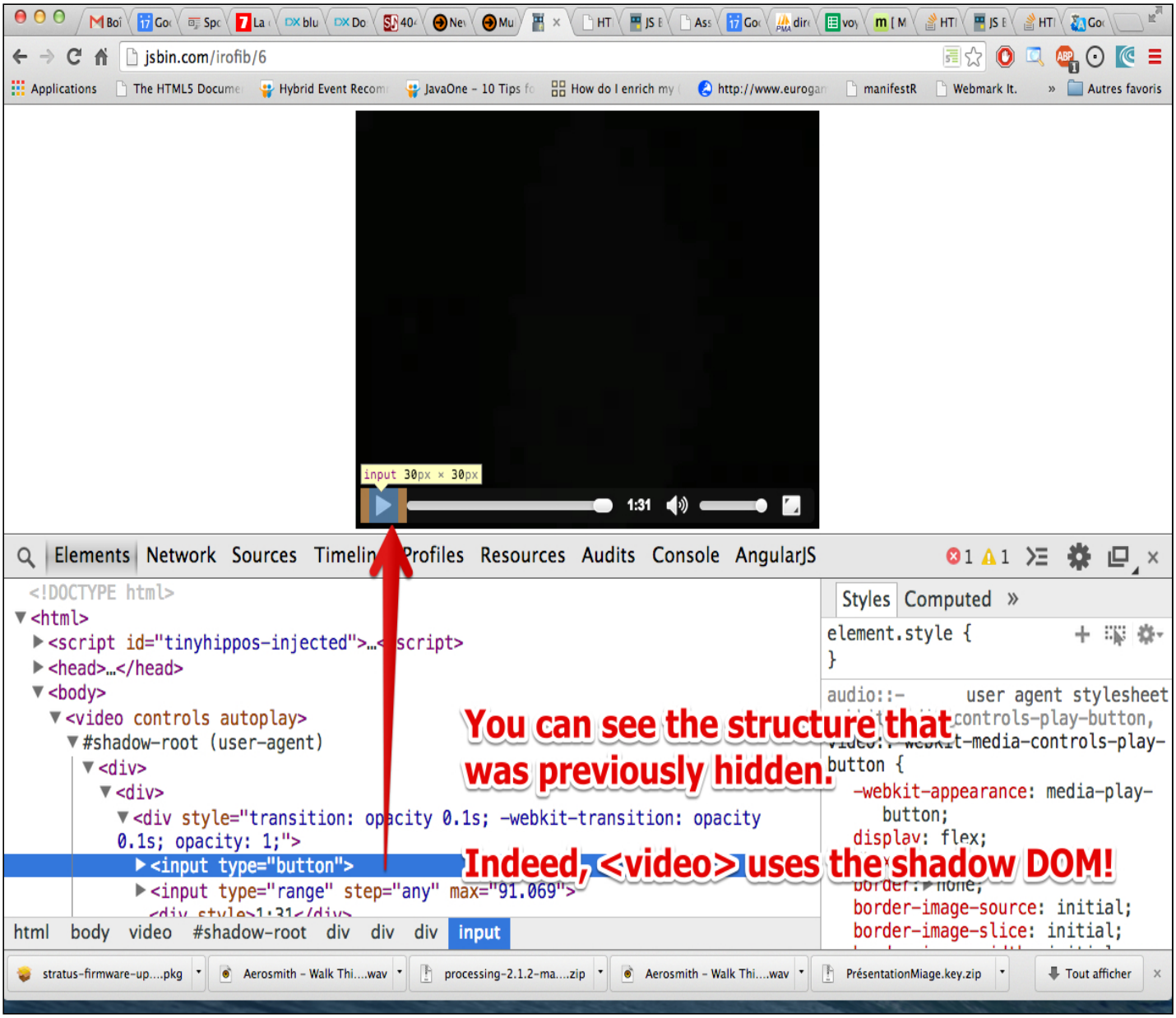

jsbin.com/irofib/6
Applications
The HTML5 Docume
Hybrid Event Recom
JavaOne - 10 Tips fo
How do I enrich my
http://www.eurogar
Autres favoris
Elements
Network
Sources
Timeline
Profiles
Resources
Audits
Console
<top frame>
<page context>
http://www.google-analytics.com/__utm.gif?utmwv=5.4
7Cutmcmd%3Dreferral%7Cutmcct%3D%2FHTML5slides%2Fi
event.returnValue is deprecated. Please use the standard event.prevent
Click here to show the options
stratus-firmware-up....pkg
Aerosmith - Walk Thi....wav
processing-2.1.2-ma....zip
Aerosmith - Walk Thi....wav
Tout afficher



Then look for the video element again within the DOM view. You should see something new:



Now, open this shadow root by clicking on it in the DOM view, and move the mouse pointer over the different elements:



Chrome developers are already using the shadow DOM to define their own Web Components, such as `<video>` or `<audio>` elements! And they use the Shadow DOM to hide the internal plumbing.

Furthermore, there is a kind of "boundary" around the `<video>`element, so that external CSS cannot interfere. The content of the`<video>` element is sort of *sandboxed* (protected from external CSS selectors, for example, or cannot be accessed using`document.querySelector()`, nor inspected by default, using a DOM inspector).

> Browser developers have been using Web Components for a while, and now

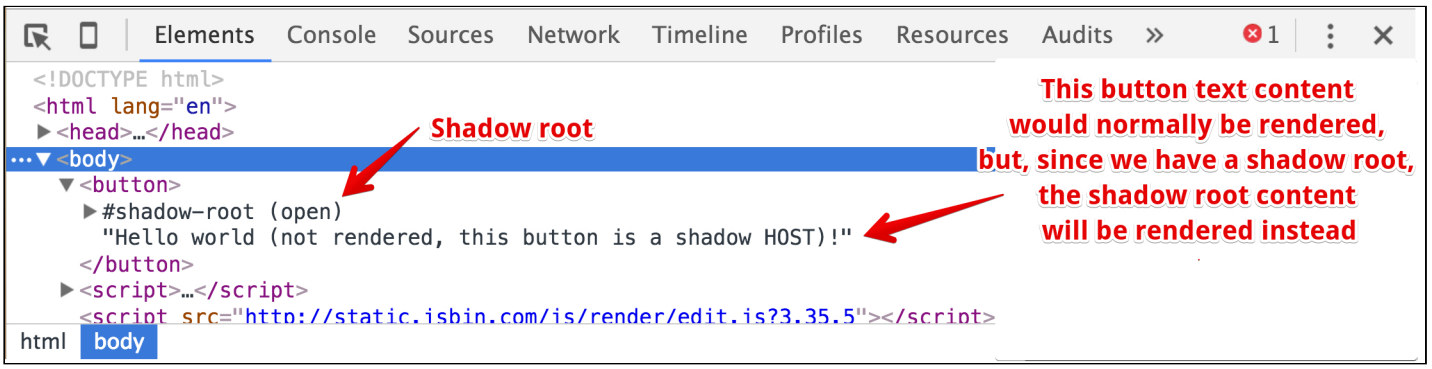
it's possible for any Web developer!

A SIMPLE EXAMPLE OF SHADOW DOM USAGE

Let's have a look at a very simple example:

```
<button>Hello, world (not rendered)!</button>
<script>
   var host =document.querySelector('button');
   var root = host.createShadowRoot();
   root.textContent = 'the shadow root node is rendered';
</script>
```

Lines 3-5 show how to associate a shadow root to an existing element. In this example, the `<button>` defined at *line 1* is a shadow host, and it is associated with a shadow root that just contains some text (*line 5*).



This example illustrates the three rules of the shadow DOM. Let's look at them again:

The three rules of Shadow DOM:

1. With Shadow DOM, elements are associated with a new kind of node: *a shadow root*.
2. An element that has a shadow root associated with it is called *a shadow host*.
3. *The content of a shadow host isn't rendered; the content of the shadow root is rendered instead.*

And indeed, the above example (try the online version here at JSBin) renders the content of the shadow root, not the content of the button. In the online example, try to change the text of the button (*line 1*), and you will notice that nothing changes. Then modify the text at *line 5* and look at the results!