Control <audio> and <video> elements from JavaScript

The <video> element has methods, properties/attributes and events that can be manipulated with JavaScript. Using the DOM API it's possible to manipulate an audio or video element as a JavaScript object that has:

- **Methods** for controlling the behavior like play(), pause(), etc.;
- **Properties** (duration, current position, etc.), either in read/write mode (like volume), or in read-only mode (like encoding, duration, etc.);
- **Events** generated during the life cycle of the element that can be processed using JavaScript callbacks. It is also possible to send events to control the video player.

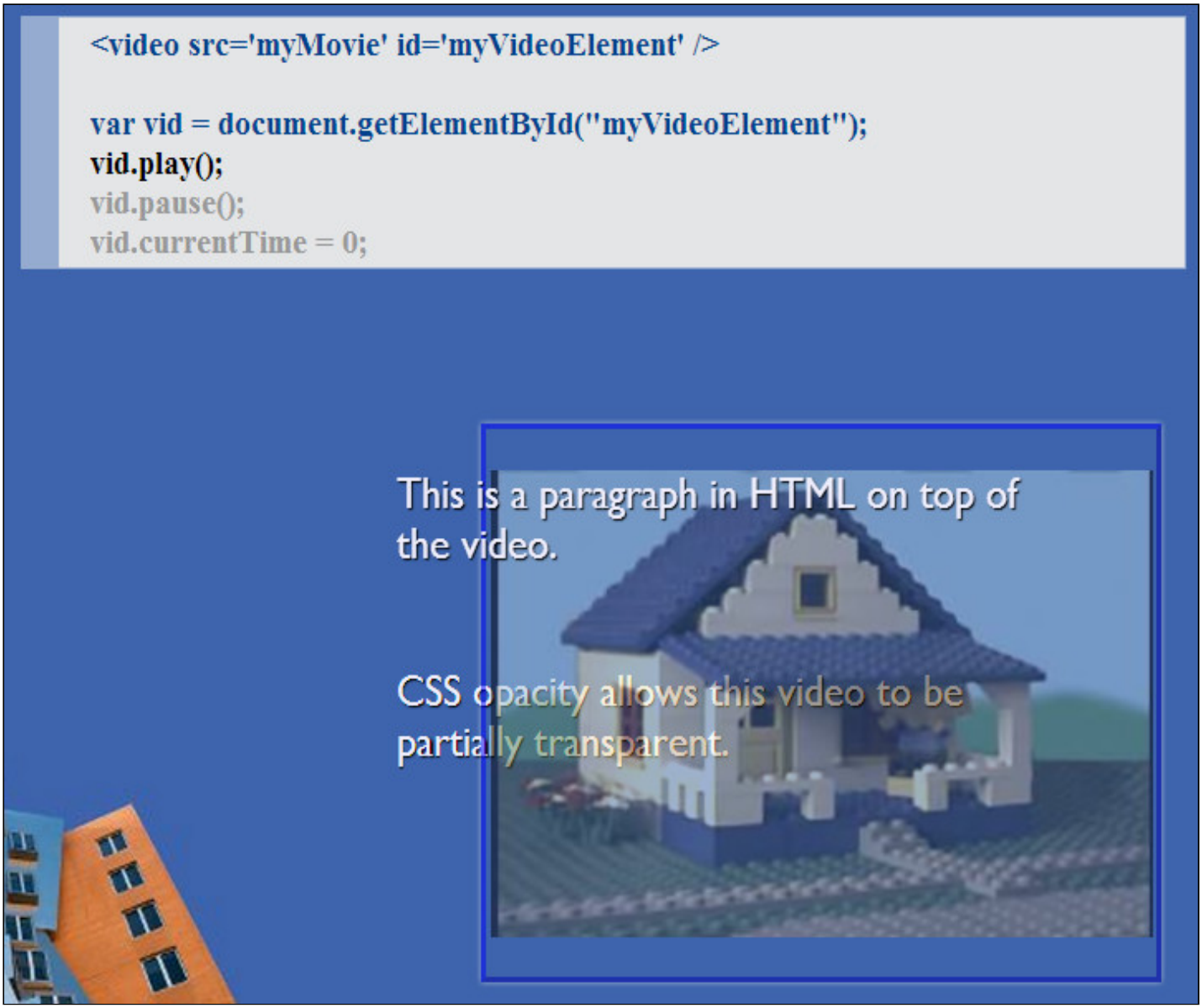
Like any HTML element, the <video> element can be manipulated/created using the DOM JavaScript API. Here is an example of programmatically creating a <video>element:

```
var video = document.createElement('video');
video.src = 'video.mp4';
video.controls = true;
document.body.appendChild(video);
```

This will create a complete video player for the file "video.mp4", with control buttons, and will add it to the <body> element of the page.

EXAMPLE THAT SHOWS HOW TO CALL PLAY/PAUSE OR REWIND A VIDEO

Please look at this interesting example originally written by P.Le Hegaret from W3C in 2008! (you can click on "edit in JS BIN" to view the source but we will give simpler and more detailed examples in the next section, this one is more to show what can be done).



Note that in order to play the video, you must click on the "vid.play()" text. To pause it, you click on the "vid.pause()" text, and so on. *Notice the text at the top of the video, as well as the transparency.* The text can be selected, since all the elements displayed are pure DOM objects. You can zoom the page in and out, etc. This was not possible with the Flash technology.

Conclusion: you can change the look and feel of the standard video player very easily: use custom CSS and design your own control widgets. We can find many examples of such video players that offer extended functionalities on the Web. We will present some of them further in the course, but before that, let's see a little more of what we can do using the JavaScript API of the `<video>` element.