Creating tracks on the fly: example with sound sprites

INTRODUCTION

In this lesson we show:

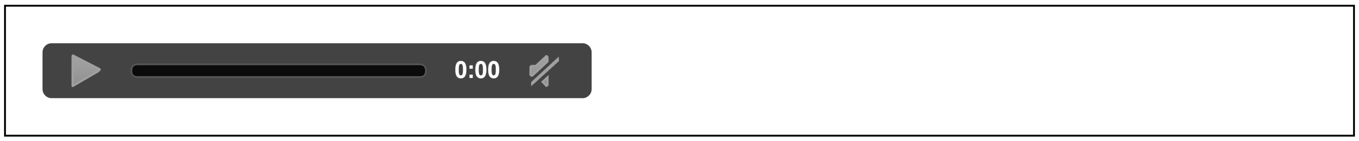
- The `addTextTrack` method for adding a `TextTrack` to an html `<track>` element,
- The `VTTCue` constructor, for creating cues: `var cue = new VTTCue(startTime, endTime, id);`
- the `addCue` method for adding cues on the fly to a`TextTrack` etc.

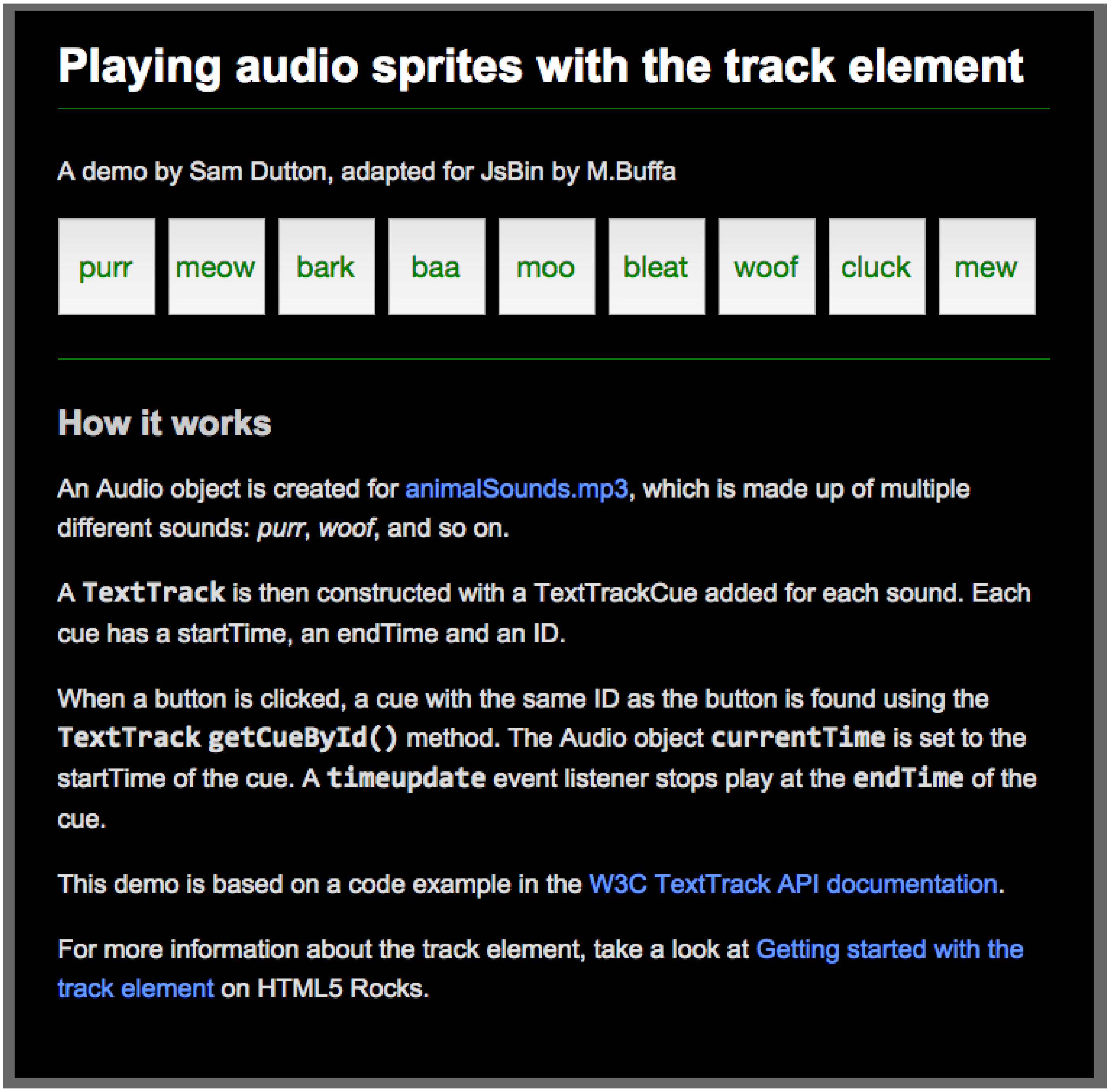
These methods will allow us to create TextTrack objects and cues on the fly, programatically.
The presented example shows how we can create "sound sprites": small sounds that are parts of a mp3 file, and that can be played separately. Each sound will be defined as a cue in a track associated with the `<audio>`element.

EXAMPLE: CREATE ON THE FLY A WEBVTT FILE WITH MANY CUES, IN ORDER TO CUT A BIG SOUND FILE INTO SEGMENTS AND PLAY THEM ON DEMAND

This JsBin demonstration, adapted from an original demo by Sam Dutton, uses a single mp3 file that contains recorded animal sounds.

Below is the sound file. You can try to play it:





Playing audio sprites with the track element

A demo by Sam Dutton, adapted for JsBin by M.Buffa

How it works

An Audio object is created for animalSounds.mp3, which is made up of multiple different sounds: *purr*, *woof*, and so on.

A **`TextTrack`** is then constructed with a TextTrackCue added for each sound. Each cue has a startTime, an endTime and an ID.

When a button is clicked, a cue with the same ID as the button is found using the **`TextTrack getCueById()`** method. The Audio object **`currentTime`** is set to the startTime of the cue. A **`timeupdate`** event listener stops play at the **`endTime`** of the cue.

This demo is based on a code example in the W3C TextTrack API documentation.

For more information about the track element, take a look at Getting started with the track element on HTML5 Rocks.

Explanations

The demo uses a JavaScript array for defining the different animal sounds in this audio file:

```
var sounds = [
   {
      id: "purr",
      startTime: 0.200,
      endTime: 1.800
   },
   {
```

```
            id: "meow",
            startTime: 2.300,
            endTime: 3.300
        },
        {
            id: "bark",
            startTime: 3.900,
            endTime: 4.300
        },
        {
            id: "baa",
            startTime: 5.000,
            endTime: 5.800
        }
        ...
];
```

The idea is to create a track on the fly, then add cues within this track. Each cue will be created with the id, the start and end time taken from the above JavaScript object. In the end, we will have a track with individual cues located at the time location where an animal sound is in the mp3 file.

Then we generate buttons in the HTML document, and when the user clicks on a button, the `getCueById` method is called, then the start and end time properties of the cue are accessed and the sound is played (using the `currentTime` property of the audio element).

HTML source code extract:

```
...
<h1>Playing audio sprites with the track element</h1>
 <p>A demo by Sam Dutton, adapted for JsBin by M.Buffa</p>

<div id="soundButtons" class="isSupported"></div>
...
```

JavaScript code extract and explanations are in the comments:

```
 window.onload = function() {
    // Create an audio element programmatically

  var audio = newAudio("http://mainline.i3s.unice.fr/mooc/animalSounds.mp3");

    audio.addEventListener("loadedmetadata", function() {
      // When the audio file has its metadata loaded, we can add
      // a new track to it, with mode = hidden. It will fire events
      // even if it is hidden
      var track = audio.addTextTrack("metadata", "sprite track", "en");
      track.mode = "hidden";

      // for browsers that do not implement the getCueById() method
      if (typeof track.getCueById !== "function") {
         track.getCueById = function(id) {
            var cues = track.cues;
            for (var i = 0; i != track.cues.length; ++i) {
               if (cues[i].id === id) {
               return cues[i];
            }
         }
      };
   }

   var sounds = [
      {
        id: "purr",
        startTime: 0.200,
        endTime: 1.800
      },
      {
        id: "meow",
        startTime: 2.300,
        endTime: 3.300
      },
      ...
   ];

   for (var i = 0; i !== sounds.length; ++i) {
      // for each animal sound, create a cue with id, start and end time
      var sound = sounds[i];
      var cue = new VTTCue(sound.startTime, sound.endTime, sound.id);
      cue.id = sound.id;
      // add it to the track
      track.addCue(cue);
      // create a button and add it to the HTML document
      document.querySelector("#soundButtons").innerHTML +=
                        "<button class='playSound' id="
                        + sound.id + ">" +sound.id
                        + "</button>";
   }
   var endTime;
   audio.addEventListener("timeupdate", function(event) {
      // When we play a sound, we set the endtime var.
      // We need to listen when the audio file is being played,
      // in order to pause it when endTime is reached.
      if (event.target.currentTime > endTime)
         event.target.pause();
   });

   function playSound(id) {
     // Plays the sound corresponding to the cue with id equal
     // to the one passed as a parameter. We set the endTime var
     // and position the audio currentTime at the start time
     // of the sound
     var cue = track.getCueById(id);
     audio.currentTime = cue.startTime;
     endTime = cue.endTime;
     audio.play();
  };
  // create listeners for all buttons
 var buttons = document.querySelectorAll("button.playSound");

  for(var i=; i < buttons.length; i++) {
     buttons[i].addEventListener("click", function(e) {
        playSound(this.id);
     });
  }
 });
};
```