JavaScript library for audio/video timeline representation

Samuel Goldszmidt
Ircam
1 Place Igor Stravinsky
75004 Paris
00 33 1 44 78 48 43
samuel.goldszmidt@ircam.fr

ABSTRACT
This short paper is a description of a JavaScript library developed for temporal media (audio or video) navigation and segmentation representation, using the HMTL5 specification.

Categories and Subject Descriptors
D.3.3 [**Programming Languages**]: Language classifications – *Javascript*

I.7.2 [**Document and text processing**]: Document preparation – *HTML, Multi/mixed media, Standards*

General Terms
Design, Experimentation, Human Factors, Standardization, Languages.

Keywords
HTML5, JavaScript, temporal media, navigation, segmentation, audio, video, representation.

1. INTRODUCTION
HTML5 provides two tags for temporal multimedia content: <*audio*> and <*video*>. These tags have a *controls* attribute that specifies whether or not to display media controls:

- play/pause buttons,
- navigation timeline to browse inside the media with a cursor,
- a display for current time and duration of the temporal media.

The HTML5 media events part can catch time update events during media playback, so JavaScript can handle it.

<*audio*> element display with controls attribute in Firefox

HTML5 also provides a *canvas* element for drawing purposes. HTML events can handle mouse events like *click*, *mouseover*, *mouseout, mousedown, mouseup* to interact with a drawing inside a canvas element.

After a partial state of the art about temporal media navigation, synchronization, annotation and segmentation, we will describe a JavaScript library, named *timeline-js* and available on github [1] which can build in any web page an interactive timeline for HTML5 media element inside a canvas element, based on previously named specifications.

WWW2012 Developer Track, April 18-20, 2012, Lyon, France.

2. STATE OF THE ART

2.1 Players
Since <*audio*> and <*video*> tags has been proposed by W3C inside HMTL5 working draft, a lot of JavaScript libraries have been released in order to build custom media players: *mediaelement.js* (mediaelementjs.com), *jplayer* (jplayer.org), *videojs* (videojs.com), *sublimevideo* (sublimevideo.net). These applications provide a way to have more control about the player graphical aspect. They often offer Flash fallback for browsers that don't support HTML5 media elements. But these libraries propose more or less just a timeline with a cursor, and play/pause buttons combined with some playlist possibilities.

2.2 Media synchronization
For media synchronization purposes, *popcorn.js* (popcornjs.org), *mugeda* (mugeda.com) and other experiments allow the user to bring together different medias on a same timeline, and export the result, a synchronization of static and temporal media (for instance a text, an image, and a sound) into a predefined web publication. These tools have been studied in our case for functionalities they offer in authoring mode, in particular for the timeline edition mode, even if in our use case, we focus on a unique temporal media annotation.

2.3 Intra-song navigation
Our goal is a bit different from representing events on an historical timeline. Even if historical timeline is time-based, we deal with a master temporal media that the user can browse and listen/view in *real-time*.

For intra-song navigation, *SoundCloud* (soundcloud.com) or *Freesound* (freesound.org) propose a waveform representation of the media (in fact an image server side processed which represents the waveform) and, for *SoundCloud* service, markers on the timeline added by users to point out interesting timecodes. This annotation aspect and navigation possibility has retained our attention. We had developed previously, in 2005, inside Semantic-Hifi European project IST-FP7 a similar intra-sound navigation widget, but based on automatic segmentation song representation (vs. manual segmentation / annotation) and using Flash technology [2].

2.4 Linking contents
We developed a demonstrator for Firefox audio tag, linking segments of audio with specific text and images [3]. This demo has been extended in *timesheets.js*, a library that proposes to rely on declarative W3C standards (namely, SMIL Timing and SMIL Timesheets) to synchronize HTML content [4]. In our demo, we wanted to explore content linked with media timecodes in a way that one temporal media could trigger all sort of rich multimedia content (and vice versa), that don't fall necessarily within a

predefined publication framework or templates, leaving open all multimedia possibilities [5]. The library only focus on temporal annotation visualization, not the semantic of the annotation or the type of content linked from these annotations; this is leave open to the user who can therefore integrate all kind of contents, for instance: show a map, a web link …

This partial state of the art regarding intra-sound (or video) navigation, has led us to develop *timeline-js* JavaScript library (currently in alpha version). *timeline-js* tries to simplify temporal media element integration and interaction with multimedia content regarding intra-media navigation and inter-media interaction.

3. *timeline-js*

3.1 Controls & navigation

When you have created via JavaScript a new instance of *timeline-js* referencing these three required values :

- *id* attribute value of a media tag element,
- *id* attribute value of a div which will contain canvas,
- duration value of the media,

the *timeline-js* GUI widget is displayed on the web page, showing in basic configuration:

- a cursor, default red,
- a timescale with some ticks for time graduation, default black background,
- an empty track, default grey background.

***timeline-js* GUI widget (default configuration)**

There are no *play*, *stop* and *pause* buttons. Theses controls can be easily set up with standard HTML markup and a small piece of JavaScript to interact with the media tag.

The widget listens to *timeupdate* event from the media element to synchronize the cursor. If the user clicks on the timeline (black background), the media element current time is set to the requested time.

With default settings, the widget represents the entire duration of the media. But the *timeline-js* instance has also scale and scroll possibilities (to zoom inside the temporal media, especially useful for long media) that can be bound to HTML5 *range input* element as seen in the figure below:

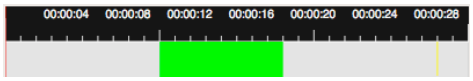
Scale and scroll range input to navigate inside media

Then, you can assemble scale and scroll range type inputs into just one user-friendly navigation bar using for instance jQueryUI range input (jqueryui.com):

Range input integration (red) for a unique navigation bar

3.2 Annotation

One use case of *timeline-js* is to represent markers and periods in a timeline to display segmentations of a media. A maker is a cue point, a single time (like the ones used in *SoundCloud*), and a period is a segment with a start time and an end time.



Period (green) and Marker (yellow)

Annotation of videos are quite similar to the annotation of sounds, it's a temporal segment of a video file. A period or a marker annotation of a video file doesn't define a shape in the video stream. This is not in the *timeline-js* scope to deal with video stream annotation, but it can be define outside, using events (cf. 3.4).

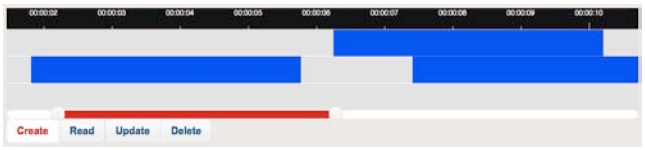
Four periods annotation of a video file

A *timeline-js* instance can also have multiple tracks (for the same media element).

Skinned version of *timeline-js* for an audio file with play/stop buttons, jQueryUI navigation bar and three tracks with different periods attached

3.3 Edition

Timeline-js has four modes: *create*, *read*, *update* and *delete*. Each mode can be set live, to change the type of interaction the user has with the interface. So far in this description, we were by default in *read* mode. In *edit* mode (see figure below), when the user clicks on the timeline, he creates a period (or a marker if *mousedown* and *mouseup* event are in the same position). In *update* mode, the user is able to update periods begin and end points. In *delete* mode, the user deletes periods or markers by clicking on it.



Create, read, update, delete mode for a three tracks timeline (switch mode buttons are designed with jQueryUI)

3.4 Events

For integration with any web application, each time an action is set on a period or a marker inside a *timeline-js* instance, it triggers a custom events:

- For periods: *createperiod*, *readperiod*, *updateperiod*, *deleteperiod*,
- For markers: *createmarker*, *readmarker*, *updatemarker*, *readmarker*.

with custom parameters, (eg. for period: new start time and end time values), and the *id* of the marker or period, allowing linking with other multimedia contents for interaction purposes or forms for edition purposes.

3.5 Miscellaneous

A configuration object can be used when instantiate the plug-in to change width, height, number of tracks, colors … A lot of other configuration parameters are available. This is intended to facilitate integration of the plug-in inside applications (these options are documented in source code). Besides, all of the configuration variables can be set live via *setOption* method on *timeline-js* instance (for instance: change mode, add track …).

4. Possible improvements

HTML5 <*track*> element could be used as a data provider for building instance of *timeline-js*. The *timeline-js* website [1] should propose soon an example.

For a better navigation experience, the scale and scroll should have an auto-scroll mode when you're zoomed in to always keep the current playback position on screen. This should be a parameter passed to the *timeline-js* instance.

SVG could be used instead of canvas to allow more custom access to the periods and markers, and express better semantic.

The Web Audio API, currently being discussed by W3C Audio Working Group, could build some interesting sound representation like waveform or spectrogram behind tracks. In the same direction, we could have automatically computed track example, for representing, for instance beats, chords, using sound features extractors already available as JavaScript libraries.

For integration with other contents, we could have a way for the user to group markers and periods together to trigger one event linked to this group. We could also have an example showing usage of media fragments working draft [6]. In this direction, when a user would click on a period, we should append `#t=period_time_in,period_time_out` to the *src* media element attribute. Last, we could try some experiments with touch event specification working draft [7] integration.

5. REFERENCES

[1] https://github.com/ouhouhsami/timeline-js/

[2] Guillaume Boutard, Samuel Goldszmidt, Geoffroy Peeters, « Browsing inside a Music Track, the Experimentation Case Study », SAMT, Athènes, 2006.

[3] http://apm.ircam.fr/page/audio-tag/

[4] Cazenave, F., Quint, V., and Roisin, C. 2011. Timesheets.js: tools for web multimedia. In *Proceedings of the 19th ACM international Conference on Multimedia* (Scottsdale, Arizona, USA, November 28 - December 01, 2011). MM '11. ACM, New York, NY, 699-702. DOI= http://doi.acm.org/10.1145/2072298.2072423

[5] Nicolas Donin, Samuel Goldszmidt, « Annoter la musique : de la segmentation de fichiers audio à la publication d'articles multimédia », *Annexes des actes d'IHM 2007* [19ème Conférence de l'Association Francophone d'Interaction Homme-Machine (Paris, France, 13-15 novembre 2007)], 2007, p. 53-56.

[6] Davy Van Deursen, Raphaël Troncy, Erik Mannens, Silvia Pfeiffer, Yves Lafon, and Rik Van de Walle. Implementing the media fragments URI specification. In *Proceedings of the 19th international conference on World wide web* (WWW '10). ACM, New York, NY, USA, 1361-1364. DOI=10.1145/1772690.1772931 http://doi.acm.org/10.1145/1772690.1772931

[7] http://www.w3.org/TR/touch-events/