HAL
open science

Timesheets.js: Tools for Web Multimedia

Fabien Cazenave, Vincent Quint, Cécile Roisin

To cite this version:

Fabien Cazenave, Vincent Quint, Cécile Roisin. Timesheets.js: Tools for Web Multimedia. ACM Multimedia 2011, Nov 2011, Scottsdale, Arizona, United States. 4 p. hal-00619396

HAL Id: hal-00619396

https://hal.science/hal-00619396v1

Submitted on 6 Sep 2011

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Timesheets.js: Tools for Web Multimedia

Fabien Cazenave, Vincent Quint
INRIA
655 avenue de l'Europe
38334 Saint Ismier, France
{fabien.cazenave, vincent.quint}@inria.fr

Cécile Roisin
Grenoble University & INRIA
655 avenue de l'Europe
38334 Saint Ismier, France
cecile.roisin@inria.fr

ABSTRACT
Timesheets.js is a JavaScript library for publishing multimedia web documents that take advantage of the new features of HTML5 and CSS3. The library allows web developers to extend their skills to synchronized multimedia contents. This technology has been experimented in a class where students had to implement an XSLT transformation for converting OpenOffice Impress presentations into web formats. The resulting slideshows run in web browsers thanks to the timesheets.js library.

Categories and Subject Descriptors
I.7 [**Document and Text Processing**]: Document Preparation—*Languages and systems, Markup languages*

General Terms
Design, Experimentation

Keywords
Declarative languages, Multimedia web applications, SMIL, HTML5, Timesheets

1. INTRODUCTION
The SMIL language has been available for a while for publishing multimedia applications on the web, but the lack of widely deployed tools has limited its impact. Among its most attractive capabilities are its timing and synchronization features, but fortunately these features are not restricted to the SMIL language and can be added to other document languages.

With the advance of HTML5, and notably its new graphic, audio and video contents, it is now possible to develop multimedia standard-based applications that can run natively in web browsers. The only missing piece is the SMIL features, that are not currently supported by browsers, but this problem can be solved by implementing these features in JavaScript. This is what we have done with the timesheet.js library.

ACM Multimedia, MM'11, November 28–December 1, 2011, Scottsdale, Arizona, USA.

With this library, sophisticated multimedia applications can be developed in a completely declarative way, and based on languages that are widely known to web developers, such as HTML and CSS. This approach is also interesting for teaching. The timing and synchronization concepts on which SMIL is based are very convenient for introducing students to the multimedia domain. We were used to present these concepts to students with the SMIL language, separately from other web languages. They can now be taught in the same context and with a more consistent approach.

The rest of the paper is organized as follows: the next section presents the timesheets.js library. Section 3 gives an example of a media annotation application that runs in the browser with this library. Finally section 4 explains how these tools are used in a course on XML and multimedia.

2. THE TIMESHEETS.JS LIBRARY
SMIL has been thought as a full specification describing all aspects of a multimedia document: content, presentation, synchronization, and interaction. However, the SMIL 3.0 Timing and Synchronization module[1] (a.k.a. SMIL Timing) is also designed to be integrated into other host languages, thus bringing synchronization and user interaction features to otherwise a-temporal document languages. As specified in this module, timing can be inserted *inline* in the markup of a static document thanks to two attributes for timing integration: `timeContainer` and `timeAction`.[2]

SMIL Timimg is complemented by another W3C specification, SMIL Timesheets,[3] that allows the most significant SMIL timing features of a document to be gathered in external resources called *timesheets*, thus separating the timing and synchronization aspects from the host language, and allowing time behavior to be shared among several documents. To paraphrase the SMIL Timesheets specification, SMIL Timing and SMIL Timesheets can be seen as a temporal counterpart of inline style and external CSS stylesheets, respectively.

2.1 Using HTML5, CSS3 and SMIL Timing
The solution we propose [1] is to combine HTML5+CSS3 and SMIL Timing/Timesheets. We take advantage of the recent addition of new media objects such as audio and video to HTML5, and new style properties such as animation and transition to CSS3. The addition of SMIL Timing extends these multimedia features significantly. It allows, for instance, some discrete parts (text, images) of

[1] http://www.w3.org/TR/smil/smil-timing.html
[2] http://www.w3.org/TR/SMIL3/smil-timing.html#q48
[3] http://www.w3.org/TR/timesheets/

a HTML page to be synchronized declaratively with the continuous parts or with other discrete parts. This also allows user interaction to be specified in a purely declarative way.

Our approach can be summed up in three points:

- use HTML5+CSS3 for structuring and styling the content and for rendering it natively in the browser with a clean content/presentation separation;
- rely on SMIL Timing/Timesheets to handle timing, media synchronization, and user interaction;
- do not redefine timing features that already exist in HTML, SVG and CSS (e.g. animations and transitions).

This approach applies to a very broad range of interactive synchronized multimedia web applications,[4] such as slide shows, captioned video clips, annotated audio recordings, graphic animations, augmented recorded conferences, interactive photo albums, web documentaries, and so on. All these applications can be developed using only declarative languages, thus making multimedia web authoring available to a broader audience. The declarative approach also provides advantages from an engineering point of view. It makes it easier to maintain and reuse content, as opposed to the purely scripting approach.

2.2 A Basic Example

As an example, here is the very simple case of a rotating banner.

```
<script type="text/javascript"
        src="timesheets.js"/>
<link href="banner.smil" rel="timesheet"
      type="application/smil+xml"/>
<div id="banner">
  <img src="image1.png"/>
  <img src="image2.png"/>
  <img src="image3.png"/>
</div>
```

where file `banner.smil` is a timesheet containing:

```
<?xml version="1.0" encoding="UTF-8"?>
<timesheet xmlns="http://www.w3.org/ns/SMIL">
  <seq repeatCount="indefinite">
    <item select="#banner img" dur="3s"/>
  </seq>
</timesheet>
```

The three images are turned by the timesheet into items displayed in a `seq`uence, each one during 3 seconds, and this sequence is repeated indefinitely.

Like in CSS, selectors are used to associate elements from the HTML document with time behaviors defined in the external timesheet. For instance, the `select` attribute of element `item` performs a `querySelectorAll()` action: for each DOM node that is matched by the `#banner img` selector, a SMIL `item` is created. This allows the same timesheet to be reused for several HTML pages: the SMIL timesheet above always works whatever the number of images in the banner.

[4] see http://wam.inrialpes.fr/timesheets/

2.3 Timesheets Engine

As SMIL Timing and Timesheets are not supported natively by web browsers, a JavaScript implementation of these specifications is required to make them available widely. Therefore, we have developed timesheets.js which is an open-source, cross-browser, dependency-free library that supports the common subset of the SMIL Timing and Timesheets specifications.

This library parses the SMIL Timing data (both inline timing and timesheet files), finds all DOM elements that are targeted (for instance, the `img` elements that are selected by `item` nodes in the `banner.smil` example above), creates a `smil` attribute on each of them with the value "idle", and schedules the activation of each target DOM element.

This scheduler acts on these DOM elements when they are activated:

- the `smil` attribute is set to "active" (if web browsers supported SMIL Timing natively, this would be a CSS pseudo-class);
- if the `timeAction` attribute is set, a specific CSS style rule is applied (e.g. "display: none", "visibility: visible", or a class is added);
- the DOM element fires a '`begin` event.

When the target element has to be deactivated, its `smil` attribute is set to "done", the specific style rule (if any) is removed, and an "end" event is fired.

This implementation obviously relies on JavaScript, but as stated above, no specific JavaScript development is required from a web developer. When an application is running, some parts of it (HTML and CSS) are executed natively by the browser, some other parts are executed by the browser's JavaScript engine.

Timesheets.js is not the first SMIL Timesheets engine running in the browser. Vuorimaa [2], for instance, has developed a Timesheets JavaScript Engine, but it was before HTML5. Therefore, it can synchronize only discrete contents.

Our implementation is available in open source under the MIT license. It is rather compact (about 2000 lines of code), and the whole engine is less than 10 Kbytes in the minified/gzipped version. There is no need to install anything for using timesheets.js; a `script` link in the HTML page to the on-line library[5] is enough (see first line of the above example).

Technically speaking, the timesheet scheduler is very modular by design:

- Each time container node has its own clock, methods, properties and event handlers.
- Each time container parses its own descendants (time nodes) and pre-computes the begin/end times according to its temporal behavior: `seq`uential, `par`allel or `excl`usive.
- All time containers expose a significant part of the HTMLMediaElement API (which is exposed by the `audio` and `video` elements of HTML5): web developers can control SMIL time containers with the usual `.play()` / `.pause()`

[5] http://wam.inrialpes.fr/timesheets/public/timesheets.js

methods, check the time with the `.currentTime` property and register to standard `timeupdate` DOM events.

3. APPLICATION: INA WEBRADIO

We have worked with INA, the French national archive of audiovisual, to publish on the web archived radio programs enhanced with associated material.[6] A typical INA Webradio page involves (see Figure 1):

- a rich audio player, i.e. a segmented timeline displaying a specific HTML fragment for each section of the audio track (bottom of Figure 1);
- buttons that users can click to display complementary content – possibly involving other multimedia sources (top right corner of Figure 1).

Figure 1: INA Webradio application

The idea is to propose a visually enhanced experience of a radio program, while allowing users to browse the content in a non-linear way, for instance with the table of contents (Sommaire) of Figure 1.

3.1 Document Processing Workflow

INA uses a SCENARI-based, XML publishing workflow to create multimedia documents. The SCENARI [7] authoring environment is used to divide a continuous media object (the recorded radio program) into several contiguous time segments, to associate an HTML fragment to each time segment, and finally to publish a dynamic multimedia document as a Flash object. One of the main limitations of this workflow, besides the usual Flash-related issues (accessibility, indexability, compatibility with mobile devices...), is that neither content authors nor web designers can control efficiently the presentation of resulting multimedia documents.

Our approach here is to keep the SCENARI content editor, but publish multimedia documents in HTML5 + CSS + SMIL Timesheets, in order to separate:

- content: every HTML fragment is defined in SCENARI;
- synchronization: SMIL Timesheets are used both to define time segments on the main audio track and to describe user interactions;
- presentation: generic CSS stylesheets can be defined by web designers for a consistent integration of these multimedia documents in the main website, and content authors can use specific style rules when necessary.

The idea here is to keep the efficiency of an XML publishing workflow and the benefits of CSS style sheets, while allowing multimedia documents to be modified in a way that is familiar to all web developers. The only specific part is the timesheet below:

```
<timesheet xmlns="http://www.w3.org/ns/SMIL">
  <!-- slide show / main section -->
  <excl timeAction="display" mediaSync="#main"
        controls="#timeController" dur="20:47">
    <item select="#section1" begin="00:00.000"/>
    <item select="#section2" begin="01:12.120"/>
    <item select="#section3" begin="04:41.742"/>
  </excl>
  <!-- extra material: multimedia pages -->
  <excl>
    <item select="#extra2"
          begin="open2.click; toc-extra2.click"
          end="close2.click; section2.end"/>
    <item select="#extra3"
          begin="open3.click; toc-extra3.click"
          end="close3.click; section3.end"/>
  </excl>
  <!-- extra material: audio -->
  <par mediaSync="#track2a" controls="#timeline2a"
       dur="2:24.039"/>
  <par mediaSync="#track2b" controls="#timeline2b"
       dur="3:59.928"/>
  <!-- extra material: rotating pictures -->
  <seq timeAction="display"
       repeatCount="indefinite">
    <item select="#extra4 img" dur="3s"/>
  </seq>
</timesheet>
```

3.2 Direct Editing

The goal of this modular approach is to enable "wysiwyg", fine-grain editing of the resulting multimedia documents.

With the initial document processing workflow, content editors could specify the semantic content for each audio section but had no easy way to refine the resulting document: that was a one-way, XML-to-Flash conversion. Every modification had to be done within the XML editor, then published in Flash to see how it looks like – and that only applied to semantic content. Now that the multimedia document is published in HTML5, it is possible to fine-tune all timing and presentation details, after conversion, directly in a Firefox extension.

Another benefit with the new workflow is that the workload can be shared efficiently between content authors and web designers: as presentation is entirely defined by CSS style sheets, web designers can work directly on the published multimedia documents to adapt the look to the website and to propose visual transitions between successive time segments, and content authors can see how all HTML fragments are displayed with these style sheets.

As the content, presentation and synchronization data are defined in three separate resources (HTML5, CSS and SMIL Timesheets,

[6] http://wam.inrialpes.fr/timesheets/public/webRadio/
[7] http://scenari-platform.org/projects/scenari/en/pres/

respectively), all local modifications can be easily backported to the XML publishing chain.

3.3 Browser Support

These HTML5+SMIL Timesheets multimedia documents are supported natively by all modern desktop browsers implementing the `audio` and `video` tags. As a consequence, users of the HTML5 version of the INA Webradio pages get a better user experience with the native multimedia player of these browsers than with external players: better responsiveness, better CPU resource usage.

Unlike the Flash version, the HTML5 Webradio pages can be served to mobile devices: most smartphones support the `audio` and `video` tags natively and can play multimedia resources without draining the battery. As the layout is defined by style sheets, a specific touch-screen layout can be served to mobile devices with a simple declarative CSS media query.

3.4 Extensibility: DOM Events, JavaScript

Most of the synchronization logic and user interaction can be defined with SMIL Timesheets. However, there are cases where a more specific dynamic interaction should be implemented.

For this purpose, each node fires a `begin` and an `end` DOM event when it is activated and deactivated by the timesheet scheduler, respectively. This allows some specific JavaScript code to be triggered easily – either with an `onbegin` / `onend` attribute in the HTML document, or with event listeners in the JavaScript code.

Besides the APIs mentioned in section 2.3 that can be used to control time containers, timesheets.js also exposes an API that allows time containers to be created dynamically. This mechanism is used in the `timesheets-controls.js` companion library to display a segmented media timeline and to keep it synchronized with the main audio track.

4. TEACHING WEB MULTIMEDIA WITH TIMESHEETS.JS

We have experienced a first course with the timesheets.js library. This course was dedicated to XML technologies and addressed XML schema definitions, the XSLT language and multimedia documents. Basic web technologies such as HTML and CSS were already known to students through previous courses. A few classes were given and then most of the work was to realize a complete project in small groups.

The goal of the project was to implement a process that: 1) merges several OpenOffice Impress presentations, and 2) transforms the result into a web document that includes navigation and animation features.

The testbed source documents contain simple slides as well as animated text and graphics. It is worth noting that the name space used by OpenOffice documents for animations is the SMIL one. It was suggested to the students to generate HTML5 content with appropriate style sheets and to reuse the `slideshow` structures and styles available on-line.[8]

Most groups succeeded in producing an animated web slideshow and proved a good understanding of both XML transformation techniques and sound structuring of multimedia documents on the web.

[8]http://wam.inrialpes.fr/timesheets/

As an example, Figure 2 shows the architecture of a project along with the generated files.

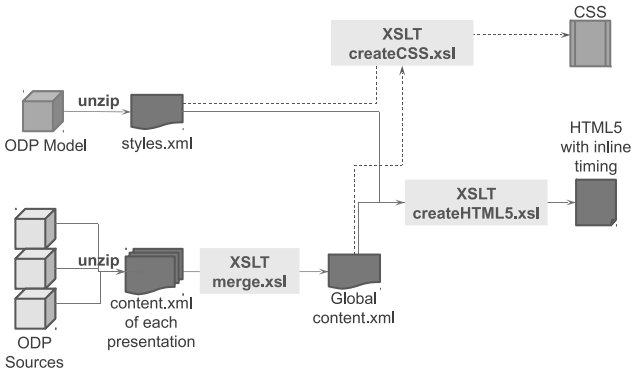

Figure 2: A student project

In the evaluation of the course, students have indicated that this project brought them a better perception of the role of the different languages of the web, including SMIL. The main difficulties they encountered were to master the XSLT language and to face the complexity of the OpenOffice style definitions.

5. CONCLUSION

We have presented the timesheets.js library and two use cases of this open technology for the web. The first one, Webradio, shows how it can be used for producing standard-based, high quality multimedia presentations, while the second one describes a teaching experience. The source code of the library and various examples are available on-line and developers are welcome to contribute on GitHub.[9]

We are currently working in two directions to 1) develop an authoring tool for media segmentation and annotation and 2) experiment microformats (such as HTML Slidy[10]) for rendering multimedia presentations with slideshow effects.

6. ACKNOWLEDGEMENTS

The work presented in this paper was done in the C2M project, funded by the French National Research Agency (ANR) under its CONTINT 2009 program. The authors are grateful to Dominique Saint-Martin from INA-GRM for providing the Webradio application.

The authors thank also Stéphane Bonhomme, the teacher of the XML course of LP SIL and the students of that class that have experimented with timesheets.js.

7. REFERENCES

[1] F. Cazenave, V. Quint, and C. Roisin. Timesheets.js: When SMIL meets HTML5 and CSS3. In *DocEng 2011: Proceedings of the Eleventh ACM Symposium on Document Engineering*. ACM, Sept. 2011.

[2] P. Vuorimaa. Timesheets JavaScript Engine, http://www.tml.tkk.fi/˜pv/timesheets/, 2007.

[9]https://github.com/fabi1cazenave/timesheets.js

[10]http://www.w3.org/Talks/Tools/Slidy2/