TOWARDS AN INTERACTIVE WEB-BASED MULTIMEDIA PLAYGROUND

Rodrigo Laiola Guimarães

IBM Research
rlaiola@br.ibm.com

ABSTRACT

In this paper we present an online code playground that pays special attention to the temporal aspect of Web documents. In particular, we consider the scenario in which modifications in a Web-based multimedia document are identified and patched in real-time, with no need to restart an ongoing presentation from the beginning. Our approach is especially useful when authoring complex Web documents containing time-based elements such as CSS3 and SVG animations, HTML5 audio and video.

***Index Terms*—** Online code playgrounds, multimedia authoring tools, immediate feedback, coding assistance, programmatic visualization, playback control, HTML, CSS, JavaScript.

1. INTRODUCTION

Online code playgrounds (e.g., W3Schools[1] and JSFiddle) have proven to be an effective mechanism of communication within computer programming forums like Stack Overflow. One of the main reasons is that such applications allow users to rapidly write, test and share proofs of concepts using just a Web browser. Nevertheless, such applications are not designed to address scenarios in which code modifications are immediately identified and applied while a multimedia presentation is running, without restarting its execution from the beginning.

In this paper we introduce Ambulant Sketchbook[2], a code playground that enables users to create and share Web-based multimedia presentations consisting of HTML (*HyperText Markup Language*), CSS (*Cascading Style Sheets*) and JavaScript code snippets. Ambulant Sketchbook is based on insights from *Inventing on Principle*[3], and it provides a simple user interface with a number of functionalities such as immediate feedback, coding assistance, programmatic visualization and playback control. Ambulant Sketchbook has been developed using many open source libraries, and currently it works in modern Web browsers like Chrome, Firefox and Safari.

[1] Some technologies mentioned in this demo paper, if unknown, could very easily be identified via a simple online search; therefore they will not be Web-referenced.

[2] Demo video available at http://goo.gl/C5OFdX.

[3] http://vimeo.com/36579366

2. AMBULANT SKETCHBOOK

A user can start creating a Web-based multimedia document immediately after accessing the application URL (*Uniform Resource Locator*) with a Web browser. Once the application Web page is completely loaded, the user visualizes a user interface (UI) as shown in Fig. 1. The UI consists of 2 main components: a code editor and a code previewer (Fig. 1.A and .B, respectively). The code editor offers independent code views for HTML, CSS and JavaScript. The user can access a particular view by simply clicking on a tab of interest (see Fig 1.1). Such code views can be used to modify specific parts of a Web-based multimedia presentation, and the effects are promptly reflected in the code previewer.

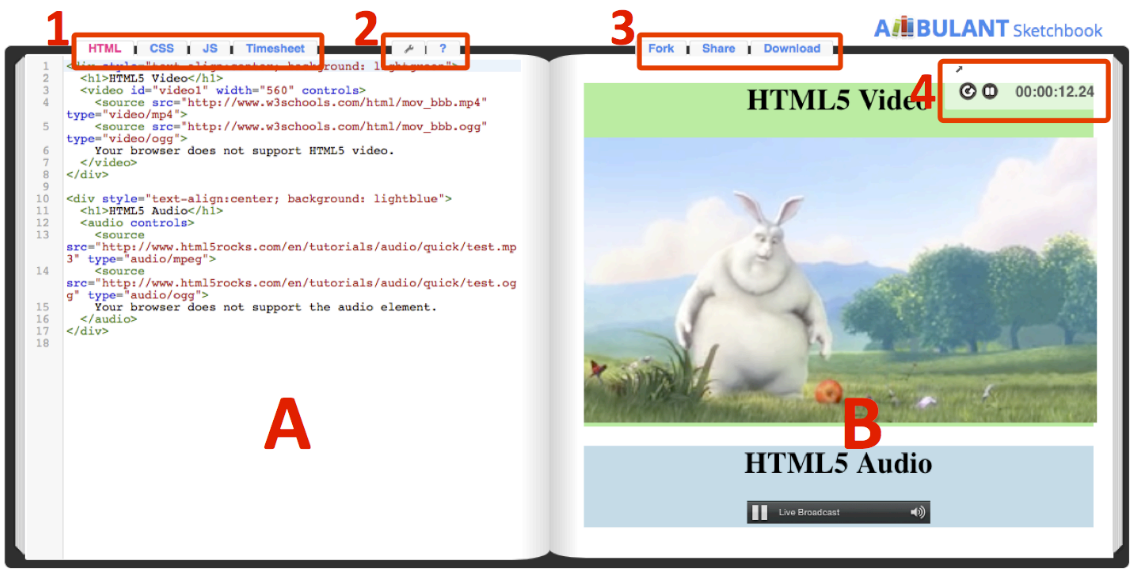
The code editor also offers some configuration options (e.g., auto play, auto update) and help support (Fig. 1.2). The user still can save the progress of a sketch to edit it later, share the URL of a sketch online (e.g., on StackOverflow), or even download it as a HTML document (Fig. 1.3). Finally, Ambulant Sketchbook provides playback primitives to control the presentation of a temporal document (Fig. 1.4). This functionality is mainly useful when working with multimedia elements like HTML5 audio and video, CSS3 and SVG animations. In the current implementation, we can play, pause, reload and visualize the presentation's elapsed time.

2.1. Architecture and implementation

Ambulant Sketchbook has been implemented and deployed using a number of Web related technologies. In the back-end, the Web server runs Apache (Linux) with PHP 5.2, and the database server runs MySQL 5.1. The front-end has been developed using a combination of server-side scripting (e.g., PHP for querying the database) and client-side languages such as HTML, CSS and JavaScript. In the client-side, we also use the jQuery framework and some third-party plugins built on top of it.

The code views (HTML, CSS and JavaScript) have been implemented using a JavaScript library that allows for in-browser code editing. As CodeMirror[4] provides only the editor component, we made use of a number of third-party add-ons for auto-completion, code hints, search etc. We also have taken advantage of CodeMirror's rich programming

[4] http://codemirror.net



```
<div style="text-align:center; background: lightgreen">
  <h1>HTML5 Video</h1>
  <video id="video1" width="560" controls>
    <source src="http://www.w3schools.com/html/mov_bbb.mp4"
type="video/mp4">
    <source src="http://www.w3schools.com/html/mov_bbb.ogg"
type="video/ogg">
    Your browser does not support HTML5 video.
  </video>
</div>

<div style="text-align:center; background: lightblue">
  <h1>HTML5 Audio</h1>
  <audio controls>
    <source
src="http://www.html5rocks.com/en/tutorials/audio/quick/test.mp
3" type="audio/mpeg">
    <source
src="http://www.html5rocks.com/en/tutorials/audio/quick/test.og
g" type="audio/ogg">
    Your browser does not support the audio element.
  </audio>
</div>
```

Fig. 1. Ambulant Sketchbook's user interface: (A) code editor; (B) code previewer; (1) tabs for different code views; (2) configuration options and help information; (3) persistence tabs; (4) playback controls.

API (*Application Programming Interface*) and CSS theming to customize the editor to our needs.

The code previewer has been implemented as an independent Web page, and embedded in the application Web page as an `<iframe>` element. This feature is particularly useful for rendering a sketch in a remote Web browser (air preview functionality). The real-time communication between the code editor and the code previewer (be this local or remote as illustrated in Fig. 2) has been implemented using the Server-Sent Events (SSE) JavaScript API. SSE is a W3C (*World Wide Web Consortium*) Working Draft that describes how servers can initiate data transmission towards clients once an initial client connection has been established. In Ambulant Sketchbook, SSEs are used to send code updates, playback commands and to activate cross-component helpers. In other words, this means that any local modification in the Web-based multimedia presentation will be reflected in real-time not only locally, but also on the remote device.

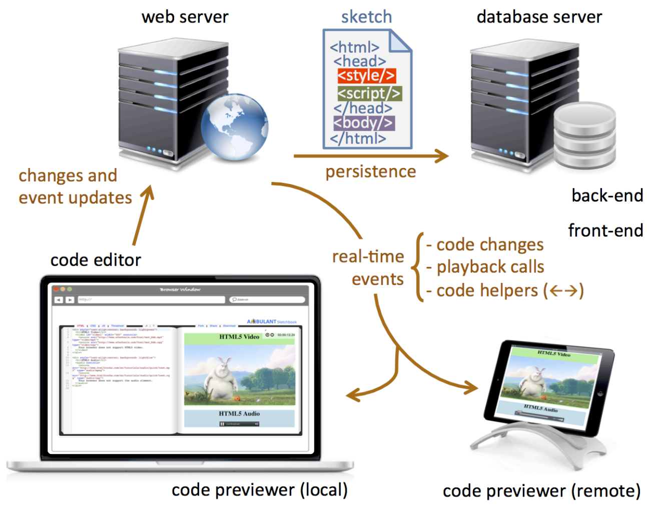


Fig. 2. System architecture.

To minimize the effect of modifications (e.g., insertions and removals) between remote DOM (*Document Object Model*) trees, we used a JavaScript *diff and patch* implementation[5]. Such library allows the abstraction of differences between DOM elements as a 'diff' object, representing the sequence of modifications that must be applied to one element in order to turn it into another element. Such diff operation is non-destructive, meaning that relocations of DOM nodes are preferred over remove-insert operations.

3. FINAL REMARKS

In this paper we present Ambulant Sketchbook, a Web-based multimedia playground in its current, yet not final, state of development. In our application, Web documents containing CSS3 and SVG animations, HTML5 audio and video can be changed and controlled in real-time without having to restart the presentation from the beginning. Preliminary results suggest that our application is a valid alternative to simplify the process of learning how to write and debug multimedia presentations on the Web [1]. As future work we intend to support other Web-based timing and synchronization mechanisms via additional code views.

4. REFERENCES

[1] R. L. Guimarães and M. M. Motta, "Design and Evaluation of an Easy-to-Use Web Playground". *Adjunct Proceedings of the 20th Brazilian Symposium on Multimedia and the Web (WebMedia '14)*, João Pessoa/PB, Brazil, November 18-21, 2014.

[5] https://github.com/fiduswriter/diffDOM