Design and Evaluation of an Easy-to-Use Web Playground

Rodrigo Laiola Guimarães* and Mateus Molinaro Motta
IBM Research
Rua Tutóia 1157
04007900 São Paulo, Brazil
+55 11 2132 3122
{rlaiola, mottam}@br.ibm.com

ABSTRACT
Learning about Web Design can be difficult and time consuming, yet students often do not learn from their errors and struggle to understand some differences between document structure, styling, scripting and temporal synchronization. In this paper we present Ambulant Sketchbook, an easy-to-use Web playground designed to enable students to understand and learn from their errors. In particular, this application simplifies the process of learning how to write and debug Web documents by exploring aspects of immediate feedback, coding assistance, direct manipulation and playback control. We have deployed and used Ambulant Sketchbook in a course of Web Design Foundations over a 2-week span. Based on the positive feedback from a group of post-secondary students, we expect the functionalities and experiences discussed in this work can yield significant insights to be considered in the design of next generation authoring tools and in the process of teaching Web Media related disciplines.

Categories and Subject Descriptors
D.2.6 [**Software Engineering**]: Programming Environments – Integrated environments, Interactive environments; I.7.2 [**Document and Text Processing**]: Document Preparation – *Hypertext/hypermedia, Languages and Systems, Standards*; K.3.1 [**Computers and Education**]: Computer Uses in Education – Computer-assisted instruction (CAI).

General Terms
Design, Experimentation, Human Factors, Languages.

Keywords
Web playgrounds, Authoring tools, Immediate feedback, Coding assistance, Direct manipulation, Playback control, HTML, CSS, JavaScript, Problem-based learning.

1. INTRODUCTION
Problem-based learning (PBL) is an instructional method that uses problems as the starting point of learning [7]. In the process, it is envisaged that students will acquire critical thinking and problem-solving skills [3]. There are varieties of PBL, but basically a typical PBL sequence consists of 1) setting up the motivational atmosphere of learning by posing an interesting problem; 2) activating learners by means of group interaction with peers and facilitators over the case; 3) building up knowledge base of relevant materials; 4) applying the knowledge to treat the case and 5) reviewing the case.

In this paper we present an application that has been designed and implemented to help using the PBL methodology to teach core concepts of Multimedia and the Web. This work is part of an extended study to better understand the intersection between human-computer interaction and multimedia authoring in online educational platforms. A group of post-secondary students enrolled in a course of Web Design Foundations has actively collaborated with this research. Starting in June 2013, students were encouraged to use existing online coding platforms to practice the concepts presented by the course instructor in the classroom. Subsequently, these students were requested to use our proof of concept, called Ambulant Sketchbook, to do the course's final assignments. After 2 weeks of interaction with the system they have been asked to provide feedback on their experiences.

2. AMBULANT SKETCHBOOK
Ambulant Sketchbook[1] is an experimental playground for rapidly prototyping *sketches* consisting of HTML (*HyperText Markup Language*), CSS (*Cascading Style Sheets*) and JavaScript[2] code snippets. The application provides a simple user interface with several useful functionalities such as immediate feedback, coding assistance, direct manipulation and playback control. Ambulant Sketchbook realizes some of the insights introduced by Bret Victor's work[3], and it has been built using many open source libraries. Currently, it works on modern Web browsers like Chrome, Firefox and Safari.

2.1 Main Functionalities
The workflow begins when a user accesses the Ambulant Sketchbook application using a Web browser. At this point, a sketch id is automatically generated and added to the URL (*Uniform Resource Locator*) as a query string. In case a valid id is provided in the initial page request, the application loads the referred sketch from the database. During the initialization, a session id is also created and this is valid as long as the Web page remains open. Once the Ambulant Sketchbook's Web page is completely loaded, the user sees a user interface (UI) like the one shown on the left side of Figure 1. The UI consists of 2 main components: the code editor and the code previewer (Figure 1-A and -B, respectively). The code editor offers independent code views for HTML, CSS and JavaScript. The user can access a

* Work was done when the author was at the Instituto Federal do Espírito Santo, Serra, ES, Brazil.

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

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

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

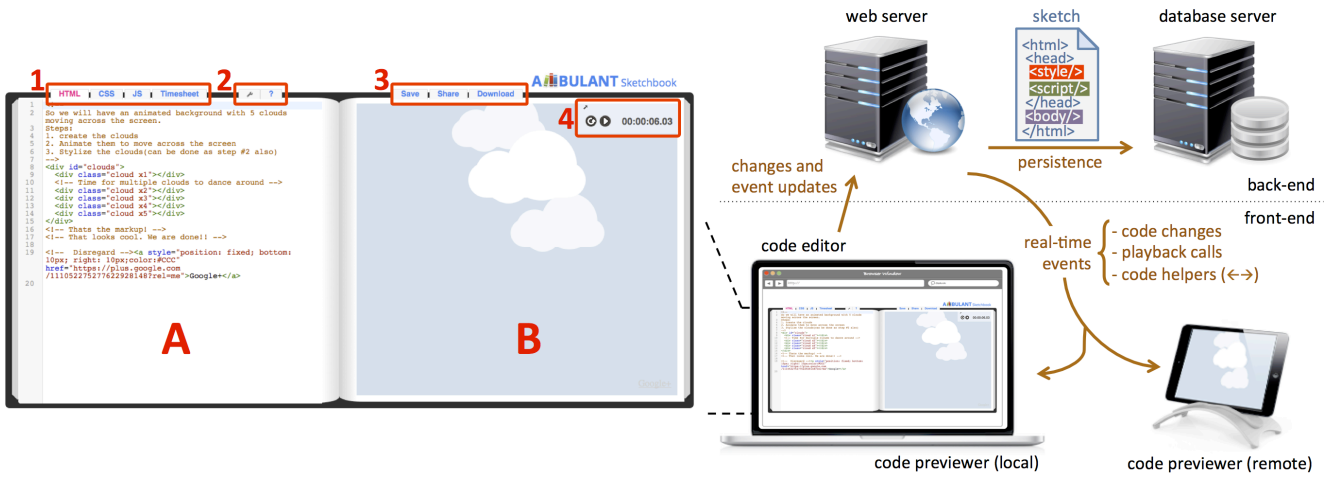

Figure 1. Ambulant Sketchbook's user interface and architecture.

particular view by simply clicking on a tab (see Figure 1-1). These views can be used to modify specific parts of a sketch, which will promptly be reflected in the code previewer.

The code editor also offers some configuration options (e.g. auto play, auto update) and help information (Figure 1-2). Moreover, depending on the part of the sketch being edited, contextual helpers can be activated with keyboard shortcuts (see Figure 2). Helpers can also be used to show the part of the code that corresponds to an element in the code previewer and vice versa.

In Ambulant Sketchbook a user can save the progress of a sketch to edit it later, share the sketch's URL online (e.g. on StackOverflow), or even download a sketch as a HTML document (Figure 1-3). In the next subsection, we will see that our application also enables the modification of a sketch locally or remotely in real-time. Finally, Ambulant Sketchbook offers playback primitives to control the presentation of a sketch (Figure 1-4). This functionality is especially useful when working with temporal elements like HTML5 audio and video, CSS3 and SVG animations. In the current prototype it is possible to play, pause, reload and visualize a sketch's elapsed time.

2.2 Implementation

Ambulant Sketchbook has been implemented and deployed using a number of Web related technologies. On the back-end, the Web server runs Apache (Linux) with PHP 5.2 while 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. On the client-side, we still used the jQuery framework and some third-party plugins built on top of it.

The code views (HTML, CSS and JavaScript) have been implemented using CodeMirror[4], a JavaScript library that allows for in-browser code editing. In principle, CodeMirror provides only the editor component, but there are a number of third-party add-ons for auto-completion, code hints, search etc. In particular, in our implementation we used an add-on[5] that offers the code helpers shown in Figure 2. A rich programming API (*Application Programming Interface*) and a CSS theming system are also available for customizing CodeMirror to fit a specific application, and extending it with new functionality. CodeMirror still offers some language-specific modes that help color (and optionally indent) text written in a particular language.

The code previewer itself has been implemented as an independent Web page, and embedded in the Ambulant Sketchbook application as an `<iframe>`. The real-time communication between the code editor and the code previewer (be this local or remote as illustrated on the right side of Figure 1) has been implemented using the Server-Sent Events (SSE) JavaScript API. SSE is a W3C (*World Wide Web Consortium*) draft describing 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.

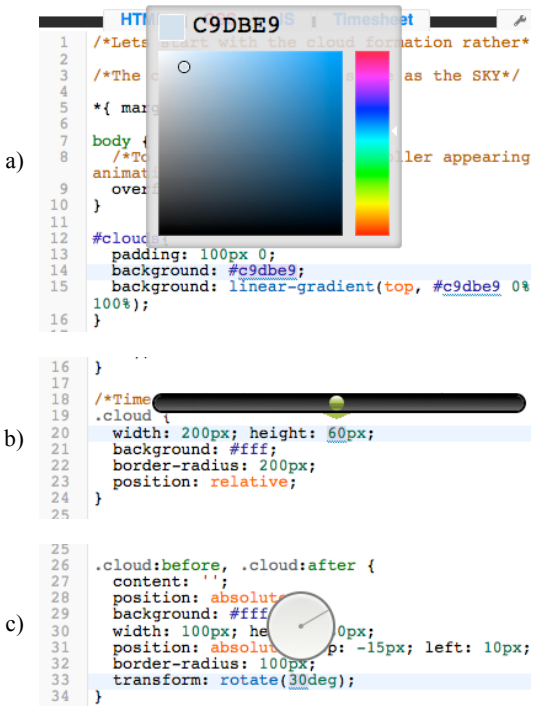

Figure 2. Contextual helpers facilitate the authoring process: a) color picker, b) slider and c) angle picker.

[4] http://codemirror.net

[5] https://github.com/enjalot/Inlet

Table 1. English and Brazilian Portuguese versions of the SUS questionnaire.

Original English version	Brazilian Portuguese version
1. I think I would like to use this system frequently.	1. Eu acho que eu gostaria de utilizar o *Ambulant Sketchbook* frequentemente.
2. I found the system unnecessarily complex.	2. Eu achei o *Ambulant Sketchbook* desnecessariamente complexo.
3. I thought the system was easy to use.	3. Eu achei o *Ambulant Sketchbook* fácil de usar.
4. I think I would need the support of a technical person to be able to use this system.	4. Eu acho que eu precisaria de suporte técnico para ser capaz de usar o *Ambulant Sketchbook*.
5. I found the various functions in this system were well integrated.	5. Eu achei que as várias funcionalidades do *Ambulant Sketchbook* foram bem integradas.
6. I thought there was too much inconsistency in this system.	6. Eu achei que havia muitas inconsistências no *Ambulant Sketchbook*.
7. I would imagine that most people would learn to use this system very quickly.	7. Eu acredito que a maioria das pessoas aprenderia a usar o *Ambulant Sketchbook* muito rapidamente.
8. I found the system very cumbersome to use.	8. Eu achei o *Ambulant Sketchbook* muito complicado.
9. I felt very confident using the system.	9. Eu me senti muito seguro usando o *Ambulant Sketchbook*.
10. I needed to learn a lot of things before I could get going with this system.	10. Eu precisei aprender muitas coisas antes que eu pudesse usar o *Ambulant Sketchbook*.

3. EVALUATION

Working in pairs, 22 post-secondary students (18 males and 4 females) from the Federal Institute of Espírito Santo (IFES) used Ambulant Sketchbook to do their assignments in the course of Web Design Foundations over a two-week span. The exercises consisted of using SVG (*Scalable Vector Graphics*) with HTML+CSS. Thereafter, students were requested to present their results to the instructor and answer a usability evaluation questionnaire. First, a short set of instructions was added that reminded them to mark a response to every statement and not to dwell too long on any one statement. All responses were anonymous. Based on our observations and responses to the questionnaires, in the remaining of this section we present the results and discuss the findings from the evaluation process.

We used the System Usability Scale (SUS) framework, which has proven to be a robust and reliable evaluation tool [4]. The original English version of the SUS questionnaire was translated into Brazilian Portuguese, as shown in Table 1. This simple ten-item questionnaire provides a single number (SUS score) that represents a composite measure of the overall usability of a system. The response to each item is selected based on a 5-point Likert scale that ranges from 1 ('Strongly disagree') to 5 ('Strongly agree'). The resulting contribution of an item ranges from 0 to 4. For positively worded items (#1, #3, #5, #7 and #9) the contribution is the scale value minus 1. For negatively worded items (#2, #4, #6, #8 and #10) the contribution is 5 minus the scale value. To calculate the overall SUS score, we sum the contributions of all 10 items and then multiply by 2.5. The final SUS score ranges from 0 to 100 (or from bad to good usability).

Figure 3 shows a graph with the distribution of the 22 SUS scores for Ambulant Sketchbook. The mean score was 90.0 (SD = 9.2), which indicates that the usability of the system was considered the *best imaginable* [2]. In principle, individual item scores are not meaningful on their own; however, Lewis and Sauro [5] show that items #4 and #10 can be used to measure the Learnability of a system at no extra cost. In our case, the resulting Learnability score was 84.7 (in a 0-100 scale), which indicates that a system like Ambulant Sketchbook is a good alternative to reduce the time it takes to complete tasks as users spend more time with it.

Participants also had the opportunity to list the advantages and shortcomings of using Ambulant Sketchbook by responding an open-ended question. On the one hand, they said that *"...it (Ambulant Sketchbook) is very useful..."* and that *"...the learning process is facilitated because we can visualize things occurring automatically..."*. Others highlighted that Ambulant Sketchbook was also fast and intuitive: *"...I particularly liked it (Ambulant Sketchbook) because it saves me time..."*, *"...I can recover from mistakes quickly..."* and *"it (Ambulant Sketchbook) makes the things I want to accomplish easier to get done..."*. On the other hand, a few students mentioned they would rather use the system in Brazilian Portuguese (if possible) and have the option to skip the first steps tutorial in the beginning of a session.

Likewise, from the instructor's point of view the overall experience was also very positive. On the one hand, Ambulant Sketchbook helped him to improve retention and performance of the students. He observed that once students got familiar with the tool they acquired more self-directed learning skills and they were motivated to solve the course assignments. On the other hand, he

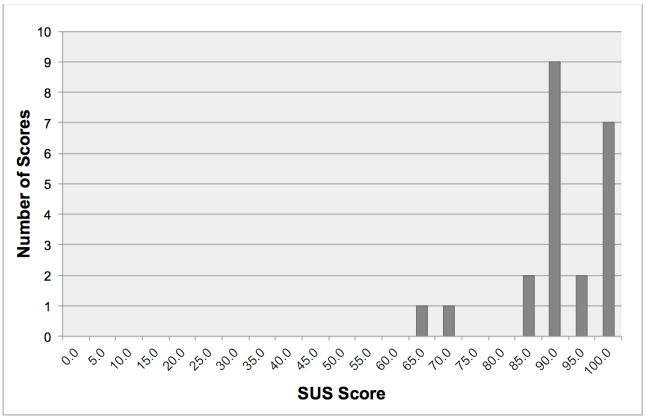

Figure 3. Distribution of the SUS scores from the data set.

pointed out that this was an ad-hoc process and a more systematic approach towards problem creation and automatic validation of PBL is desirable (e.g. use of badges). We share the same view and we think this is a topic for further work.

4. RELATED WORK

Online code playgrounds allow users to rapidly write, test and share code using just a Web browser. These tools (e.g. W3Schools[6], JSFiddle[7], JSBin[8] etc.) provide many capabilities that focus on rapid prototyping and generation of proofs of concept. There are evidences that the use of code playgrounds in communities like StackOverflow is an effective mechanism to help users solving problems [6]. Despite the functionalities online code playgrounds offer, most of them do not keep the runtime state between code changes. In this work we go a step further by considering the change of a Web document in real-time, but still preserving its previous state. Currently, this works for changes in the CSS code. Moreover, we provide a play/pause functionality that controls the presentation of temporal elements all together (e.g. CSS3 and SVG animations, HTML5 audio and video). Last but not least, Ambulant Sketchbook allows one to visualize and control the presentation of a sketch on a remote device.

Related to this work, Elm[9] is a functional reactive programming language designed to simplify the creation of rich and responsive Web browser based user interfaces. Although Elm takes some of the features discussed in this paper to the next level, it uses a programming paradigm that is not aligned to current Web practices. Our work is related but we focus on the task of supporting HTML, CSS and JavaScript.

In the field of online education, it is worth mentioning platforms like Code Academy and Khan Academy, which offer tutorials to learn a programming language. Students enrolled in a course are able to follow a series of steps at their own pace. These platforms are closely related to ours, but they only consider 'canned' exercises that can be easily validated programmatically. For more complex PBL problems this approach does not seem to scale. Moreover, these platforms are targeted to distance learning, and do not consider the intrinsic dynamics in the classroom.

5. FINAL REMARKS

In this paper we presented Ambulant Sketchbook, a Web playground developed as a flexible testbed to investigate the use of PBL while teaching Web Design principles. Based on our evaluation process, we discussed the results and provided some insights that can be considered in the design of next generation authoring tools targeted to the education domain. In particular, this work highlights the importance of considering:

- *Immediate feedback*: rapid and incremental feedback allows users to make fewer errors and complete tasks in less time, because they can see the results of an action instantly;
- *Coding assistance*: an authoring tool should provide helpers that ease the coding process [1] (e.g. completion, syntax highlight, color picker, angle picker);
- *Direct manipulation*: users should be able to interact and manipulate (e.g. dragging, resizing) the representation of objects of interest in a more intuitive way and such actions should reflect directly in the source code; and
- *Playback control*: as time now plays a prominent role on the Web, it is important to offer mechanisms to control and simulate the behavior of elements over time.

In the near future we will make Ambulant Sketchbook's source code available, although we are still deciding on its open source license. As future work, we expect to improve the current implementation and explore the use of XML (*eXtensible Markup Language*) *diff and patch* to minimize the effect of modifications on the structure of the HTML document. Finally, we intend to add support to SMIL (*Synchronized Multimedia Integration Language*) Timesheets snippets via an additional code view.

6. ACKNOWLEDGMENTS

This work has been partially funded by the Brazilian Ministry of Science and Technology under contract no. FINEP 03.11.0371.00. Special thanks to the students from the Federal Institute of Espírito Santo who have been involved in the evaluation process.

7. REFERENCES

[1] Azevedo, R. G. A., Lima, B. S., Soares Neto, C. S. and Teixeira, M. M. 2009. An approach for textual authoring of hypermedia documents based on the use of programmatic visualization and hypertextual navigation. In *Proceedings of the XV Brazilian Symposium on Multimedia and the Web* (WebMedia '09). ACM, New York, NY, USA, 8 pages. DOI=10.1145/1858477.1858495 http://doi.acm.org/10.1145/1858477.1858495

[2] Bangor, A., Kortum, P., and Miller, J. 2009. Determining What Individual SUS Scores Mean: Adding an Adjective Rating Scale. *Journal of Usability Studies* 4, 3 (May 2009), 114–123.

[3] Baturay, M. H. and Bay, O. F. 2010. The effects of problem-based learning on the classroom community perceptions and achievement of web-based education students. *Journal Computer Comput. Educ.* 55, 1 (August 2010), 43-52. DOI=10.1016/j.compedu.2009.12.001 http://dx.doi.org/10.1016/j.compedu.2009.12.001

[4] Brooke, J. 1996. SUS: A "Quick and Dirty" Usability Scale. In Jordan, P. W., Thomas, B., Weerdmeester, B. A., and McClelland, A. L., editors, *Usability Evaluation in Industry*, 189–194. Taylor & Francis, London.

[5] Lewis, J. R. and Sauro, J. 2009. The Factor Structure of the System Usability Scale. In *Proceedings of the 1st International Conference on Human Centered Design: Held as Part of HCI International 2009* (HCD '09), Masaaki Kurosu (Ed.). Springer-Verlag, Berlin, Heidelberg, 94-103. DOI=10.1007/978-3-642-02806-9_12 http://dx.doi.org/10.1007/978-3-642-02806-9_12

[6] Sillito, J., Maurer, F., Nasehi, S. M. and Burns, C. 2012. What makes a good code example?: A study of programming Q&A in StackOverflow. In Proceedings of the 2012 IEEE International Conference on Software Maintenance (ICSM) (ICSM '12). IEEE Computer Society, Washington, DC, USA, 25-34. DOI=10.1109/ICSM.2012.6405249 http://dx.doi.org/10.1109/ICSM.2012.6405249

[7] O'Grady, M. J. 2012. Practical Problem-Based Learning in Computing Education. *Trans. Comput. Educ.* 12, 3, Article 10 (July 2012), 16 pages. DOI=10.1145/2275597.2275599 http://doi.acm.org/10.1145/2275597.2275599

[6] http://www.w3schools.com

[7] http://jsfiddle.net

[8] http://jsbin.com

[9] http://elm-lang.org