openHTML: Designing a Transitional Web Editor for Novices

Thomas H. Park
College of Info. Science & Tech.
Drexel University
3141 Chestnut Street
Philadelphia, PA 19104 USA
thomas.park@drexel.edu

Ankur Saxena
College of Info. Science & Tech.
Drexel University
3141 Chestnut Street
Philadelphia, PA 19104 USA
ankur.saxena@drexel.edu

Swathi Jagannath
College of Info. Science & Tech.
Drexel University
3141 Chestnut Street
Philadelphia, PA 19104 USA
swathi.jagannath@drexel.edu

Susan Wiedenbeck
College of Info. Science & Tech.
Drexel University
3141 Chestnut Street
Philadelphia, PA 19104 USA
susan.wiedenbeck@drexel.edu

Andrea Forte
College of Info. Science & Tech.
Drexel University
3141 Chestnut Street
Philadelphia, PA 19104 USA
andrea.forte@drexel.edu

Abstract

We describe the initial design rationale and early findings from studies of a web editor for beginners called openHTML. We explain our strategy of *transitional design* that views web editors as a part of a complex socio-technical system that spans multiple tools, practices, and actors. Our goal is to create a toolkit that can engage beginners in meaningful activities now and prepare them for more sophisticated activities in the future.

Author Keywords

Learner-Centered Design; Web Development; Code Editors.

ACM Classification Keywords

H.5.2. Information interfaces and presentation (e.g., HCI): User Interfaces: User-centered design.

General Terms

Design, Human Factors.

Introduction

HCI as a field has long been concerned with designing technologies for novice users. Diverse approaches to “learnable” interface design have been explored; for example, advanced features can be hidden and complexity can be simplified to help new users ease into sophisticated software and tasks [4]. We are exploring the effectiveness of a *transitional design* to support novice web developers. Our web editor, openHTML, allows users to author web pages with minimal setup and training but without shielding them from the code, allowing them to become familiar with the syntax and semantics of HTML and CSS.

The web is a ubiquitous computational environment, where people from non-technical backgrounds often develop technical skills and knowledge [5] to meet varied needs—from a student customizing their blog, to a business owner requiring an online presence [12]. Basic web development tasks can cause anxiety and frustration for newcomers. Blackwell notes that markup languages share many pitfalls of programming: “As with the use of JavaScript, even the abstractions of HTML provide the opportunity for syntax errors, runtime errors, or bugs in the form of unintended or exceptional behaviors” [2]. Beginners in our preliminary studies often found HTML and CSS challenging. Misconceptions about basic concepts such as markup tags and hyperlinks were common, and quickly stymied their progress in learning to build websites [10]. In order to transition from these basic tasks to sophisticated ones, beginners need support in making sense of early experiences with web development. This calls for improved understanding of the difficulties they experience and how they might successfully be overcome.

Transitional Design

Our approach to designing the openHTML environment takes inspiration from the instructional design literature. Educational technologists are deeply concerned with supporting novices, and *scaffolding* is a commonly cited approach. Scaffolding has two goals: “(1) to enable learners to achieve a process or goal which would not be possible without the support and (2) to facilitate learning to achieve without the support” [6]. Examples of scaffolded environments include intelligent tools that track students’ activities and intervene with feedback when needed and tools that structure processes and elicit articulation. These are examples of “within-tool” scaffolding—temporary support for activities that is carefully designed into a tool. We are exploring within-tool scaffolding that could help novice web developers.

It is also important to recognize “between-tool” scaffolding as a design strategy for supporting novices. Puntambekar and Kolodner note that scaffolding is not necessarily a feature of a single tool; rather, it can be distributed throughout a socio-technical system [11]. We view openHTML as one part of a larger system of tools and practice that includes not only the immediate development context, but also the tools and practices that learners may have access to as their skills become more advanced. In other words, openHTML itself can be thought of as a *transitional design*—support that fades through disuse when the learner no longer needs it or wants to accomplish tasks that require a more sophisticated tool.

We are using an iterative, learner-centered process [13] to design a web editor that supports novices in creative exploration of web building and prepares them to use more sophisticated editors should they choose to do so. In the next sections, we describe early design iterations and three rounds of evaluation.

Initial Design of a Web Editor

In this section, we provide a rationale for the design of openHTML. Our aim for the initial prototype was to provide support for novice web developers as they became familiar with HTML and CSS syntax. Our design was guided by the following principles.

1. Coding is a priority. This provided a useful design constraint by helping us decide where complexity could be hidden and where it should be exposed. Web development encompasses diverse activities, including non-coding activities like setup of the development environment and server management, which can be perceived by novices as barriers to making web pages [10]. We integrate web hosting with the editor, reducing the steps between delving into the code and deploying it online.

2. Iterative learner-centered design. We are interested in exploring how a web editor can be designed to support learning and teaching; consequently, we consider factors like motivation and understanding. These considerations not only underscore the importance of providing the low barrier to coding activities described above, but also compelled us to avoid a WYSIWYG interface that conceals syntax. We started with a basic code editor and are studying it in multiple contexts to discover what kinds of support beginners need to use such an environment effectively.

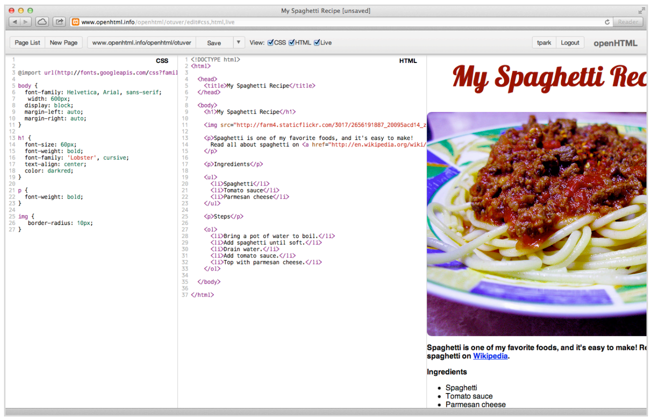
3. Sharing and audience are critical. Education literature suggests that sharing and discussing interesting creations is central to the learning experience. In the design of openHTML, we want to support people in not only making and displaying web creations, but also in sharing, discussing, and reusing their source code with one another.

openHTML is a browser-based editor, meaning no installation is needed to start using it. We built it by modifying an open-source tool used for JavaScript debugging called JS Bin [7].

The main edit screen presents three panes: editing CSS, editing HTML, and previewing the rendered page (Figure 1). The edit panes provide basic syntax highlighting and line numbering. By juxtaposing CSS with HTML and giving them equal visual prominence, users are encouraged to experiment with CSS and use it in concert with HTML to style their pages. The preview updates instantaneously as the user edits code in the other panes.

openHTML eliminates the need to organize files locally and upload them to a remote server. Once a page has been saved, it is immediately accessible to others online. The code of a web page can also be shared with other users, who can then make their own copy and add edits to it. This feature makes it possible to remix others' work and for instructors to seed a webpage for an assignment or tutorial.

Clicking the "Page List" button takes users to a list of previously created web pages (Figure 2). All revisions of a web page are saved and can be accessed here; as in a wiki system, saving revisions reduces the risk associated with experimentation because previous versions can always be restored. Users can also join a group, giving another user like an instructor an organized overview of the group's pages.


My Spaghetti Recipe [unsaved]
www.openhtml.info/openhtml/otuver/edit#css,html,live
Page List
New Page
www.openhtml.info/openhtml/otuver
Save
View: CSS HTML Live
tpark
Logout
openHTML
CSS
HTML
My Spaghetti Rec

```css
@import url(http://fonts.googleapis.com/css?famil

body {
  font-family: Helvetica, Arial, sans-serif;
   width: 600px;
  display: block;
  margin-left: auto;
  margin-right: auto;
}

h1 {
  font-size: 60px;
  font-weight: bold;
  font-family: 'Lobster', cursive;
  text-align: center;
  color: darkred;
}

p {
  font-weight: bold;
}

img {
   border-radius: 10px;
}
```

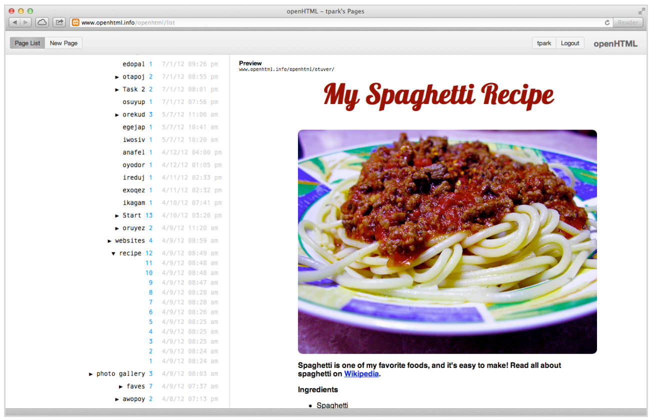
```html
<!DOCTYPE html>
<html>

  <head>
    <title>My Spaghetti Recipe</title>
  </head>

  <body>
    <h1>My Spaghetti Recipe</h1>

    <img src="http://farm4.staticflickr.com/3017/2656191887_20095acd14_z

    <p>Spaghetti is one of my favorite foods, and it's easy to make!
      Read all about spaghetti on <a href="http://en.wikipedia.org/wiki/
    </p>

    <p>Ingredients</p>

    <ul>
      <li>Spaghetti</li>
      <li>Tomato sauce</li>
      <li>Parmesan cheese</li>
    </ul>

    <p>Steps</p>

    <ol>
      <li>Bring a pot of water to boil.</li>
      <li>Add spaghetti until soft.</li>
      <li>Drain water.</li>
      <li>Add tomato sauce.</li>
      <li>Top with parmesan cheese.</li>
    </ol>

  </body>

</html>
```

Spaghetti is one of my favorite foods, and it's easy to make! Re spaghetti on Wikipedia.

Ingredients

- Spaghetti
- Tomato sauce
- Parmesan cheese

Figure 1: Editing a page in openHTML with all three panes open, from left to right: CSS, HTML, and preview.



```
        edopal  1   7/1/12 09:26 pm
      ▸ otapoj  2   7/1/12 08:55 pm
      ▸ Task 2  2   7/1/12 08:01 pm
        osuyup  1   7/1/12 07:56 pm
      ▸ orekud  3   5/7/12 11:06 am
        egejap  1   5/7/12 10:41 am
        iwosiv  1   5/7/12 10:20 am
        anafel  1   4/12/12 04:00 pm
        oyodor  1   4/12/12 01:05 pm
        ireduj  1   4/11/12 02:33 pm
        exoqez  1   4/11/12 02:32 pm
        ikagam  1   4/10/12 07:41 pm
       ▸ Start  13  4/10/12 03:20 pm
      ▸ oruyez  2   4/9/12 11:20 am
    ▸ websites  4   4/9/12 08:59 am
      ▾ recipe  12  4/9/12 08:49 am
                11  4/9/12 08:48 am
                10  4/9/12 08:48 am
                 9  4/9/12 08:47 am
                 8  4/9/12 08:28 am
                 7  4/9/12 08:28 am
                 6  4/9/12 08:26 am
                 5  4/9/12 08:25 am
                 4  4/9/12 08:25 am
                 3  4/9/12 08:25 am
                 2  4/9/12 08:24 am
                 1  4/9/12 08:24 am
▸ photo gallery  3  4/9/12 08:03 am
       ▸ faves  7   4/9/12 07:37 am
      ▸ awopoy  2   4/8/12 07:13 pm
```

Spaghetti is one of my favorite foods, and it's easy to make! Read all about spaghetti on Wikipedia.

Ingredients

- Spaghetti

Figure 2: openHTML "page list" view where users can select a page to edit or expand a time-stamped list of all revisions for each page.

Finally, openHTML has been instrumented with fine-grained logging for research purposes. Keystroke-level edits are recorded and can be used to reconstruct and play back coding sessions.

Evaluations

We have conducted three rounds of evaluation: a pilot test with children at a community center, a lab study with adult users of different skill levels, and a field study of novices in a university course. In this extended abstract, we briefly describe the lab study.

We conducted the lab study to investigate the errors people make while coding with HTML and CSS, and the role played by the web editor and other resources in making and resolving these errors. We recruited 20 participants via paper fliers, Craigslist ads, and postings on public mailing lists, and gave them twenty dollars for their time. Participants had diverse backgrounds, with 5 females and 15 males ranging from 18 to 47 years of age (M=24.4, SD=7.9). All had experience with HTML, and all but one CSS; additionally, 18 of the participants had some experience with a programming language such as JavaScript or PHP.

Participants were invited to the Drexel University iSchool usability lab and asked to complete five HTML and CSS coding tasks that resemble assignments in introductory web development courses. Tasks were presented in order of increasing difficulty. For instance, participants were asked to create an ordered list and sub-list in the first task, a hyperlink and embedded image in the second task, and a content area and sidebar in the final task. For each task, participants were given a set of instructions and a screenshot showing how the final web page should appear.

The participants were asked to follow a think-aloud protocol articulating their thought process as they completed the tasks. They were instructed to complete the tasks to the best of their ability, using web searches if necessary. As participants completed the tasks, audio, video, and actions on the screen were captured using the software Morae Recorder. The task study was followed with a brief interview about participants' experiences using openHTML. Sessions lasted between 32 and 91 minutes (M=60.0, SD=15.9)

To understand the challenges that beginner web coders face and how the web editor supported or failed to support their efforts, we are using thematic analysis [3] to identify common errors and recovery activities across participants. Thematic analysis is an inductive and iterative process of identifying meaningful patterns in data that is well suited for making sense of rich, semi-structured data. We have begun the coding process by examining all the data and identifying conceptual themes. Through successive rounds of analysis, we will classify coding errors and their underlying causes, as well as recovery strategies and their effectiveness. This will help us determine where our next iteration of design efforts will have the greatest impact.

Preliminary Findings

In this section, we present themes and some design implications based on our first round of analysis.

Tinkering

When attempting to debug coding errors, participants often engaged in tinkering — making repeated modifications to a piece of code in quick succession. Whether the tinkering indicated playful and productive "bricolage" [1] or aimless "thrashing" was informed by the context in which it occurred. For example, tinkering between multiple valid CSS values demonstrated design exploration; a web editor might facilitate this through direct manipulation of a range of values, rather than repeatedly typing in values. In contrast, tinkering with CSS property names was often symptomatic of a breakdown in understanding the semantics of CSS; a web editor in this case might discourage tinkering and offer a different form of support.

Web Searches

Participants regularly turned to web searches when trying to resolve coding errors. Web searches were extremely code-centric and action-oriented. Participants frequently ignored explanatory text when using web resources to understand and correct an error, instead preferring to navigate directly to code snippets, paste them into their own code, and manipulate them within the code editor. These findings suggest an opportunity to integrate explanations more closely with code in these resources, or even further, integrate explanations within the web editor itself.

Latent and Active Errors

Active errors are errors that can be immediately perceived by the user, while latent errors give no immediate cue, only manifesting later due to interaction with other conditions [8]. In our analysis, we have found it useful to consider the coding errors made by participants in these terms, as well as when the participant recognized an error has been made and which part of the interface provided cues.

With the first iteration of openHTML, active errors expressed primarily in the live preview (e.g., a syntax error causing a broken image), although some participants also relied on the syntax highlighting (e.g., unclosed tags). On the other hand, latent errors, which did not express in openHTML's interface, occurred often and would sometimes cascade into a series of additional errors. While some are syntax errors that can be detected through validation, others require techniques such as the uniqueness heuristic, where a class or ID used only once throughout a project may indicate an error [9]. Through the design of openHTML, we are finding new ways of bringing these latent errors to the surface.

Conclusion

This work describes our initial design of openHTML and preliminary findings from our evaluation. openHTML acts as a "between-tools" scaffold, allowing beginners to quickly build web pages and develop coding skills in before transitioning to more sophisticated and complex tools. It also serves as a platform for introducing "within-tools scaffolding", through features that support their learning as novices. Our findings highlight a number of these design opportunities. Through ongoing research, we will continue exploring the role that a transitional web editor can play in giving beginners positive, productive learning experiences.

Acknowledgements

This work is supported by NSF grant 1152094. We are grateful for their support.

References

[1] Beckwith, L., Kissinger, C., Burnett, M., Wiedenbeck, S., Lawrance, J., Blackwell, A., & Cook, C. Tinkering and gender in end-user programmers' debugging. In *Proc. CHI 2006*, 231-240.

[2] Blackwell, A. First steps in programming: A rationale for attention investment models. In *Proc. HCC 2002*, 2–10.

[3] Braun, V. & Clarke, V. Using thematic analysis in psychology. *Qualitative Research in Psychology*, *3*, 2 (2006), 77–101.

[4] Carroll, J. & Carrithers, C. Training wheels in a user interface. *CACM, 27*, 8 (1984), 800-806.

[5] Dorn, B. & Guzdial, M. Discovering computing: Perspectives of web designers. In *Proc. ICER 2010*, 23-29.

[6] Guzdial, M. Software-realized scaffolding to facilitate programming for science learning. *Interactive Learning Environments, 4*, 1 (1995), 1-44.

[7] JS Bin. http://jsbin.com/

[8] Ko, A. & Myers, B. A framework and methodology for studying the causes of software errors in programming systems. *Journal of Visual Languages and Computing*, 16 (2005), 41–84.

[9] Ko, A. & Wobbrock, J. Cleanroom: Edit-time error detection with the uniqueness heuristic. In *Proc. VL/HCC 2010*, 7-14.

[10] Park, T. & Wiedenbeck, S. Learning web development: Challenges at an earlier stage of computing education. In *Proc. ICER 2011*, 125-132.

[11] Puntambekar, S. & Kolodner, J. Toward implementing distributed scaffolding: Helping students learn science from design. *Journal of Research in Science Teaching*, *42*, 2 (2005), 185–217.

[12] Rosson, M., Ballin, J., & Nash, H. Everyday programming: Challenges and opportunities for informal web development. In *Proc. VL/HCC 2004*, 123-130.

[13] Soloway, E., Guzdial, M., & Hay, K. Learner-centered design: The challenge for HCI in the 21st century. *Interactions*, *1*, 2 (1994), 36–48.