Quantifying and Reducing the Cost of Web Edits

Edward Benson
MIT CSAIL
Cambridge, MA 02139 USA
eob@csail.mit.edu

Abstract

The web is an increasingly important medium of expression for both technical and non-technical authors, and the family of web languages (HTML, CSS, Javascript, *etc*) is poised to become the *de facto* content authoring format across disciplines. For such a future to become possible, the ease with which authors—especially non-technical authors—can express themselves becomes paramount. This work presents a method of reducing the complexity of source-level HTML authoring and shows its experimental impact on usability. Our initial findings suggest a path toward simpler HTML source organization without sacrificing rich, modern design.

Author Keywords

Web design; Web content editing; HTML

ACM Classification Keywords

H.5.4. [**Information Interfaces and Presentation**]: Hypertext/Hypermedia - User Issues

Introduction

The family of web languages (HTML, CSS, JS, *et al*) are increasingly becoming the *de facto* format for content authorship. In addition to the tremendous growth of the web for both publication and application development, authors are increasingly using web languages to write content for other platforms. All three major eBook formats [1] make heavy use of HTML and CSS. Math rendering libraries and multi-column text support push web languages closer to being a viable alternative to $\LaTeX$for academics, with interactive Javascript charts as an added bonus. And elementary schools, led by the Partnership for 21^{st} Century Skills, are increasingly asking students to submit writing assignments as blog posts and wiki pages [10] instead of on paper.

WYSIWYG authoring tools are an important part of this growing ecosystem, but raw, source-level editing is also an essential skill. Web languages were designed, after all, with text-mode source editing in mind. No matter how sophisticated our WYSIWYG tools, there will always be circumstances where an author with knowledge of HTML and CSS is advantaged over one at the mercy of a graphical tool.

A major hindrance to source-level HTML usability is the complexity of the HTML being written. Plain, content-centric HTML (`<h1>`, `<p>`, `<ul>`, `<li>`, *etc*) is light-weight and allows the author to focus on the content. Modern web pages, by contrast, are steeped in complex presentational HTML scaffolding necessary to support the CSS which styles the page. Our previous study found this design scaffolding accounts for as much as 27% of the raw bytes of a web page and a much higher percentage of the tag complexity, since content is mostly text while the scaffolding is mostly tags [2].

[1] EPUB, IBA (iBooks), and KF8 (Kindle)

This work proposes and evaluates a new method of HTML source organization that mitigates the rising complexity in HTML documents. The goal of this method is to boost the usability of source-level HTML editing to support a future in which HTML remains a convenient format to edit by hand for both technical and non-technical users.

The method, which we call *mockup stitching*, consists of separating an HTML document into two documents: one design mockup and one content document. A third document contains a set of simple relations which relate regions in one document to regions in the other. Mockup stitching enables content authoring and design authoring to be completely partitioned, and as a result, content authoring can be performed using simpler HTML than today's techniques would permit.

Mockup stitching is not simply web templating, though there are clearly parallels. Both the mockup and the content document are coherently renderable web pages, unlike the template paradigm. And the document which stitches them is external to both, like CSS. This enables the design mockup to even be a live site hosted elsewhere.

Our evaluation consists of a user study which finds that the simplified document structure made possible by mockup stitching results in significantly less cost for content edits. This suggests that mockup stitching would be a useful practice to adopt across the many communities which use web languages as their content format.

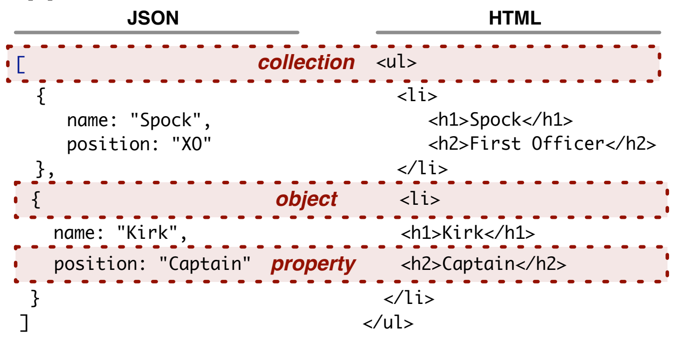
Related Work

Prior work investigates the cost of web authoring from a project management perspective [8]. We instead focus on how the source-level authoring *method* affects the cost. Methods to improve web authoring usability generally fall into one of three categories: language extensions, sensemaking tools, and automation.

Language extension methods focus on expanding the basic vocabulary or syntax of existing languages. Vocabulary examples include the `video` tag (HTML5) and rounded corners in CSS3. Syntax approaches such as HAML [3] and Markdown [6] provide a simpler way to describe web documents, but they fail to address the co-mingling of design and content concerns within the HTML document. XML can provide such a separation, but it require the user to resort to XSLT, a complex, non-HTML authoring format. We enable the author to achieve this separation of concerns without leaving HTML.

Sensemaking approaches help the user better understand the workings of a web document. WebCrystal [4] and FireCrystal [9], for example, help authors identify the source code that causes a web fragment to appear or behave a certain way. Finally, automation approaches enable the user to manipulate low-level HTML by using a simpler, higher-level interface. WYSIWYG editors are the canonical example, and some have argued that such tools are necessary for high-quality hypertext authoring [11]. More recent work focuses on assisting the task of web design retargeting: CopyStyler [5] provides interactive support with an editor interface that places pages side-by-side, and Bricolage [7] uses machine learning to automate the process.

Approach



```
[                                    <ul>
  {                                    <li>
    name: "Spock",                       <h1>Spock</h1>
    position: "XO"                       <h2>First Officer</h2>
  },                                   </li>
  {                                    <li>
    name: "Kirk",                        <h1>Kirk</h1>
    position: "Captain"                  <h2>Captain</h2>
  }                                    </li>
]                                    </ul>
```

Figure 1: The content page in our method uses the minimum possible HTML structure to attain an isomorphic mapping to JSON. The user can then express data as HTML—a language intended for hand editing—instead of another, less editor-friendly format, such as JSON.

Our approach is to split HTML documents into two separate and self-contained documents: a design mockup and a content document. The *design mockup* contains all the complex HTML scaffolding and Javascript code necessary to achieve the desired content design. It could either be an actual page, live on the web (a university department might maintain a standard academic homepage mockup for example), or a privately maintained mockup.

The *content document* is what the casual observer might describe as a “1993-style” web page: a simple, bare-bones HTML document containing just content. This document contains the minimum possible HTML structure that enables an isomorphic mapping from HTML to a JSON object: an element for each collection, each semantic object, and each object property, as shown in Figure 1. For a blog post, this might be three HTML elements: one to represent the blog post object, and one to contain the title, and one to contain the post body.

We relate this simplified content document to the mockup using a simple CSS-like relational language called Cascading Tree Sheets (CTS) described in [1]. CTS allows relations to be made between documents using CSS selectors that allow a mockup to be used as if it were a template, and the content document to be used as if it were a data source for that template. The end result is that the author is able to create and edit web content using the simplified content document, and these edits will have an effect at run-time on the fully decorated, design-laden page.

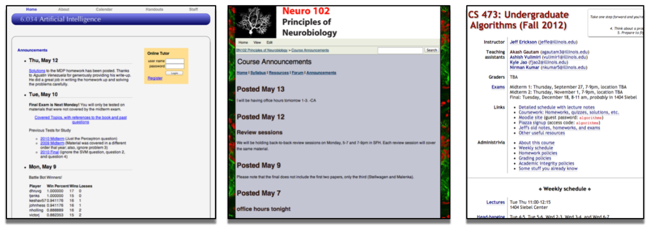


Figure 2: Screen shots of the three course pages used in our study.

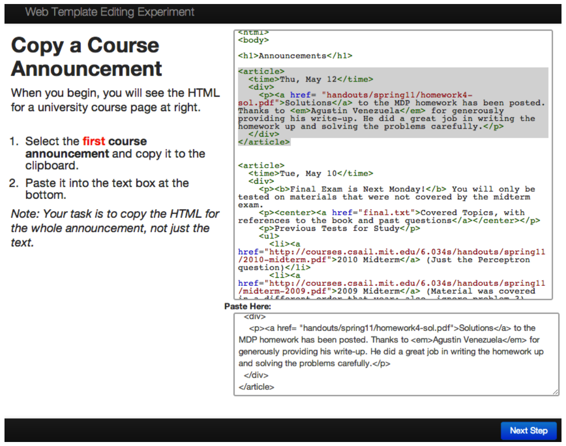
Our method maintains an ascetic adherence to backwards and forwards compatibility with current practice: it uses nothing more than ordinary HTML pages with a Javascript runtime to stitch them together. Notably, this means that our method can leverage the existing body of approaches to HTML authorship. Because content documents, for instance, are ordinary HTML, they can be authored in WYSIWYG editors.

Study and Evaluation

We evaluate this workflow with a user-study that compares this mockup stitching approach to ordinary source editing. Participants were asked to copy, paste, and edit course announcements on a university class page. We chose the copy, paste, and edit operations because they are the quintessential primitive tasks of anyone maintaining source-level web content, and we chose university class pages because they are a typical (and familiar) domain in which by-hand HTML editing frequently occurs.

Our study consisted of 17 computer professionals (both graduate students and industry professionals) who program for a living, all of whom reported having had experience authoring in HTML before. For each action (copy, paste, edit), subjects performed a practice task and then two timed tasks, for a total of 12 data points per user. Subjects were randomly grouped into two blocks, which differed in the order in which methods were presented to the subject. Block one performed tasks using traditional HTML editing first (which we'll call the **HTML** method) and block two performed tasks using the simplified content document first (which we'll call the **HTML-Simple** method).

Our data set was created from three actual course announcement pages from university courses, shown in Figure 2. The HTML method utilized the raw HTML from these pages. The HTML-Simple method utilized a hand-created copy of each course page in which just the announcements were extracted and the HTML structure minimized to just four HTML elements per announcement (a wrapper, title, time, and body).



Copy a Course Announcement

When you begin, you will see the HTML for a university course page at right.

1. Select the **first** course **announcement** and copy it to the clipboard.
2. Paste it into the text box at the bottom.

Note: Your task is to copy the HTML for the whole announcement, not just the text.

```
<html>
<body>

<h1>Announcements</h1>

<article>
  <time>Thu, May 12</time>
  <div>
    <p><a href= "handouts/spring11/homework4-
sol.pdf">Solutions</a> to the MDP homework has been posted.
Thanks to <em>Agustin Venezuela</em> for generously
providing his write-up. He did a great job in writing the
homework up and solving the problems carefully.</p>
  </div>
</article>


<article>
  <time>Tue, May 10</time>
  <div>
    <p><b>Final Exam is Next Monday!</b> You will only be
tested on materials that were not covered by the midterm
exam.
    <p><center><a href="final.txt">Covered Topics, with
references to the book and past questions</a></center></p>
    <p>Previous Tests for Study</p>
    <ul>
      <li><a
href="http://courses.csail.mit.edu/6.034s/handouts/spring11
/2010-midterm.pdf">2010 Midterm</a> (Just the Perceptron
question)</li>
      <li><a
href="http://courses.csail.mit.edu/6.034s/handouts/spring11
/midterm-2009.pdf">2009 Midterm</a> (Material was covered
```

Paste Here:

```
  <div>
    <p><a href= "handouts/spring11/homework4-sol.pdf">Solutions</a> to the
MDP homework has been posted. Thanks to <em>Agustin Venezuela</em> for
generously providing his write-up. He did a great job in writing the homework up
and solving the problems carefully.</p>
  </div>
</article>
```

Figure 3: Copy task using the **HTML-Simple** method. The only difference to the **HTML** method is the HTML the subject must navigate.

The experiment was automated as a web application which guided participants through each step. For each discrete task, the subject was presented with a briefing of the steps they would perform, followed by the HTML source for that task, which was syntax-highlighted and line-wrapped to improve readability. For all tasks, we recorded the time between the user clicking a “Begin Task” button (after reading the briefing) and clicking a “Finished” button. Task descriptions were:

Copy Task. Each copy task asked a user to copy either the first, second, or third course announcement from the HTML source and paste it into a text box. Example shown in Figure 3.

Paste Task. Before starting the paste task, users were given an HTML fragment to copy to the clipboard. They were then asked to paste that fragment after the first, second, or third course announcement in the HTML source that was revealed.

Edit Task. Users were asked to change the title of an announcement. To do this, they had to find the announcement in the HTML source and then edit it.

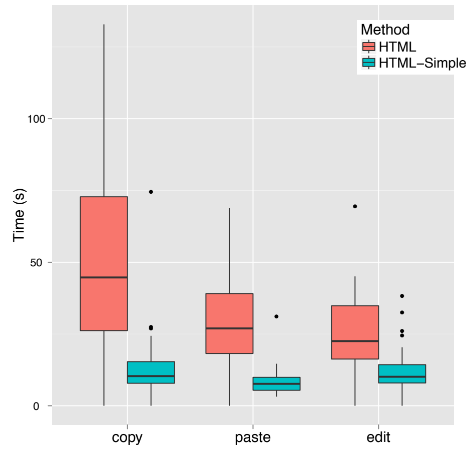


Figure 4: Whisker plot results for the two methods on copy, paste, and edit actions. Measurements are in seconds and dots represent outliers.

We find that for all tasks, the simplified editing approach results in faster edits, as depicted in Figure 4. A two-way within subjects analysis of variance found this interaction between the method and the completion time to be significant: $F(1,2) = 23.89$, $P < 0.001$. Table 1 records mean time to completion for each method and action.

These results suggest that communities that edit HTML would benefit from dividing their HTML authoring workflow into separate design- and content-centric documents.

	HTML	HTML-Simple
Copy	53	**13.6**
Paste	31.1	**8.5**
Edit	25.6	**12.6**

Table 1: Mean time to completion, in seconds, for each method and action type.

Future Work

We are currently extending this work to attain a fuller picture of web authoring usability in two ways. First, we are adding to the set of tasks that we test. This experiment tested basic editing tasks, but we are also interested in: (1) **widget inclusion**, in which we plan to measure the usability trade-offs between Javascript APIs and microformat-based APIs, and (2) **retargeting cost**, in which local content is modified to match style from elsewhere.

We are further extending our editing study to control for various degrees of HTML complexity. Our current approach measures a near maximum signal-to-noise ratio against current practice; we plan to programatically vary this ratio so that we can fit a curve to the edit cost as a function of HTML design complexity.

References

[1] Benson, E., and Karger, D. Cascading Tree Sheets and Recombinant HTML: Better Encapsulation and Retargeting of Web Content. In *WWW* (2013).

[2] Benson, E., Marcus, A., Karger, D., and Madden, S. Sync kit: a persistent client-side database caching toolkit for data intensive websites. In *WWW* (Apr. 2010).

[3] Catlin, H., and Weizenbaum, N. Haml.

[4] Chang, K. S.-P., and Myers, B. A. WebCrystal: understanding and reusing examples in web authoring. In *CHI* (May 2012).

[5] Fitzgerald, M. J. CopyStyler : Web design by example. *MIT Masters Thesis* (2008).

[6] Gruber, J. Markdown language specification.

[7] Kumar, R., Talton, J. O., Ahmad, S., and Klemmer, S. R. Bricolage: example-based retargeting for web design. In *CHI* (2011).

[8] Mendes, E., Watson, I., Triggs, C., Mosley, N., and Counsell, S. A comparative study of cost estimation models for web hypermedia applications. In *Empirical Software Engineering* (2003).

[9] Oney, S., and Myers, B. FireCrystal: Understanding interactive behaviors in dynamic web pages. In *VLHCC 2009* (2009).

[10] Partnership for 21st Century Skills. Framework for 21st Century Skills, 2012.

[11] Thimbleby, H. Gentler: A Tool For Systematic Web Authoring. In *International Journal of Human-Computer Studies*, vol. 47 (1997).