Noel Reed Fiser. CSS Spa: A System for Parsing & Aggregation of Cascading Style Sheets. A Master's Paper for the M.S. in I.S degree. April, 2007. 48 pages. Advisor: Brad Hemminger

Cascading Style Sheets (CSS) represent a powerful means for Web designers to separate HTML content from presentation. CSS solves many design-related issues, but it also creates new ones, including document management issues and "code bloat."

The CSS Spa has been created to assist designers with CSS management. After spidering a Web site's CSS content, the CSS Spa follows a series of rules to accomplish this: enforcement of atomic separation; elimination of non-CSS style elements; elimination of inline styles; normalization attributes and values; creation of new CSS "superclasses;" and finally removal of any unused CSS.

The CSS Spa tool was developed using PHP and MySQL, then tested on two sets of sites: three simple learning cases and three real-world Web sites. Each of these test sites represents a different CSS challenge: reliance on older HTML styles; heavy use of inline styles; and enhancement of an already-efficient design.

Headings:

Web sites -- Design.

Cascading style sheets.

Computer graphics.

Web publishing.

# CSS SPA: A SYSTEM FOR PARSING & AGGREGATION OF CASCADING STYLE SHEETS

by
Noel Reed Fiser

A Master's paper submitted to the faculty
of the School of Information and Library Science
of the University of North Carolina at Chapel Hill
in partial fulfillment of the requirements
for the degree of Master of Science in
Information Science.

Chapel Hill, North Carolina

April 2007

Approved by

Brad Hemminger

# TABLE OF CONTENTS

## TABLES AND FIGURES

## LIST OF ABBREVIATIONS

| | |
|---|---|
| AV | attribute-value |
| CPP | classes per page |
| CSS | Cascading Style Sheets |
| CSS Spa | CSS System for Parsing and Aggregation |
| DOM | Document Object Model |
| HTML | HyperText Markup Language |
| MSIE | Microsoft Internet Explorer |
| SGML | Standard Generalized Markup Language |
| *TSA* | *Technology Source Archives* |
| WWW or Web | World Wide Web |
| XHTML | Extensible HyperText Markup Language |
| XML | Extensible Markup Language |

For a father who encourages the practical,
A mother who fosters the fanciful,
A step-father who inspired the actual,
And a family who pursues the ideal.

I never intended HTML source code to be seen by users. A browser/editor would let a user simply view or edit the language of a page of hypertext, as if he were using a word processor. The idea of asking people to write the angle brackets by hand was to me, and I assumed to many, as unacceptable as asking one to prepare a Microsoft Word document by writing out its binary coded format.

- Tim Berners-Lee, 1999

In fact, it has been a constant source of delight for me over the past year to get to continually tell hordes (literally) of people who want to -- strap yourselves in, here it comes -- control what their documents look like in ways that would be trivial in TeX, Microsoft Word, and every other common text processing environment: “Sorry, you're screwed.”

- Marc Andreessen, co-founder of Netscape, 1994

## Introduction

On August 6, 1991, the World Wide Web (WWW or Web) was created, when Tim Berners-Lee published the first Web “site.” His vision of the Web was one of an editable, interactive community where people could share ideas. To this end, he adapted his invention of hypertext, i.e. text that links to other text, into Hypertext Markup Language (HTML), which would become the foundation language of the Web.

In these early days, pages were rather primitive and HTML was rather simple, providing only the barest amount of style on top of a Web page’s information. But as more people began to play with HTML, something unexpected happened: the Web quickly moved beyond the original intentions of its creator. Beyond adding graphics, which was a standard hypertext feature, people began using HTML to lay out their pages in a rich, robust, and artistic fashion by means of what might be deemed a "hack": the <TABLE/> tag, officially proposed in May 1996 but made available much earlier in the Netscape Navigator version 1.1.[1] "What Web designers quickly discovered was that tables were a powerful way to control, to some extent, the position of elements on a page. Tables with invisible borders could act as transparent templates, in which designers aligned various elements, like headers and footers." (Weiss, 2006). Suddenly the Web as we know it was born. No longer confined to strictly defined layouts, creativity on the Web began to flourish and casual users (and therefore commercial interests as well) began to look at the Web as a tool not just for information but also for entertainment.

As Web activity rose exponentially, so did this unforeseen reliance on table layout. As a result, in practice Web developers were using HTML—which was intended to be a content-rich delivery system—to bury data inside of design. So most every Web page had a major percentage of its bulk devoted to display issues. In direct response to this phenomenon, the World Wide Web Consortium (W3C), also founded by Berners-Lee, proposed Cascading Style Sheets (CSS) in December 1996.

Stylesheets had been in common use since the 1970's in Standard Generalized Markup Language (SGML), the predecessor to HTML. What they offered here was a method of separating the messy union of content and display that early Web HTML had quickly become. Thus CSS allows a Web developer (or in fact a *team* of developers) to create one document for information and a second for how the information should be presented. This separation, in combination with the unique contribution of CSS—the

[1] W3C RFC 1942, May 1996. Netscape Navigator 1.1, April 1995.

"cascade or inheritance of attributes from a parent node to its children in the Document Object Model (DOM)—gave CSS a strong footing in the Web development industry, and it is now an indispensable tool for Web page design.[2]

## *The benefits of CSS*

Cascading Style Sheets offer some basic benefits to both Web developers and Web users alike. Some of these are enumerated here:

- CSS demands a fundamental separation of presentation from content;
- CSS can accommodate many different presentation layers (e.g., high-resolution screens, lower-resolution screens, printers, etc) and user disabilities with the least effort;
- this separation allows designers and developers to work in parallel, creating a more streamlined Web page development process;
- stylesheets that can be cached result in faster page downloads;
- CSS support is non-intrusive, allowing older browsers to ignore its presence and still present a legible Web page; and
- hierarchical, cascading effects on page elements result in fewer explicit stylistic inclusions; and
- CSS implicitly encourages the reuse of presentation code and thus can also produce a more consistent user experience.

[2] In fact, CSS version 1 was adopted rather slowly and even though CSS version 2 has been in officially published since May 1998, it was never fully adopted because of varying browser implementations. See Koch, 2007, for a complete chart of cross-browser adoption: http://www.quirksmode.org/css/contents.html.

*Separation of presentation from content.* As already mentioned, a primary reason for the advent of CSS was the need to separate data and semantic structure from presentation and stylistic concerns. “This separation is a requirement for device-independent documents (all device-specific information is left to the stylesheet) and simplifies document management, since a stylesheet can describe many documents” (Lie & Saarela, 1999). In order to accomplish this, CSS is most often stored in an entirely separate text-based “stylesheet” document (usually provided the .css extension). But CSS can also be applied from within the <HEAD/> HTML tag on the same page or inline as an STYLE attribute on any relevant tag. This flexibility allows developers to affect CSS changes in any number of locations. And it initially offered a relatively smooth transition from classic, inline HTML style tagging to the inline STYLE tagging of CSS, since the new CSS code could remain in the same location where the old styling code had resided.

Furthermore, one can link multiple media-specific CSS documents to a single HTML page (e.g., screen, print, mobile, etc). This allows the page’s data to remain intact and consistent across multiple outputs as well as visitor types (e.g. the disabled). Only the choice of stylistic representation is necessary to manage.

*Working in parallel.* The isolation of the presentation layer from content also allows the development of focused specialists in a team environment. Visual designers can isolate style issues in documents and use CSS to manage them, while data specialists and systems programmers can produce HTML code that is largely style-independent. This functional separation streamlines project development as well as allowing more individual iterations through the software development process.

Problems such as loss of time and investment can quickly arise from a design team working in a more serialized environment. Holly Nardini learned this first-hand during the 2001 redesign of the Yale University Library Web site:

> After receiving the coded draft Web pages, we learned that the designers had contracted the coding to an outside programmer. While we had asked for "clean" HTML, what we received included an abundant and unnecessary use of nested tables, single-pixel images used for spacing, and JavaScript. Because we received the coded pages so late in the process and were working on a deadline, we didn't have the luxury of the iterative process that we'd had with the designers. We were able to go back to the programmer only twice with requests for changes. When we ended up recoding much of the pages ourselves, the design remained intact with far simpler code. Perhaps if the programmer had been included in all our meetings with the designers, we would have been able to communicate our needs better and received code that required less reworking on our part (2002).

*Faster Web sites.* Because CSS can be (and often is) stored in external files, these static files do need to be downloaded only once during a visit to a Web site. So the server may deliver many HTML documents during a session, but all the style information was contained in a single CSS document downloaded once. While this savings does not affect page delivery times as much as imagery might, the bandwidth gain becomes increasingly significant as more and more pages are viewed.

*A subtle but powerful footprint.* CSS embeds itself in HTML as inline attributes, <STYLE/> tags, or external <LINK/> files. The latter two are most often embedded in the <HEAD/> of an HTML page, otherwise hiding them from the visitor. Each of these insertion mechanisms will be ignored by older or less capable browsers. Thus, CSS minimally interferes with the data on the page; when CSS is disabled the data is still legible and useful.[3]

*Hierarchy and cascade.* Beyond offering simply stylistic options, the other, equally-important function of CSS is the cascade, meaning that hierarchy and order matter in CSS. Rich style information can be applied to any element on a page with a single line of CSS code. But hierarchy allows developers to break those original "rules" by applying a particular class or identifier to a sub-element. This cascading effect requires much less code and effort than applying a stylistic effect in every location where it might be desired, simplifying the developer's job immensely. It can also make page- or site-wide style changes almost trivial.

*A holistic view.* Conversely, the cascading nature of CSS helps to encourage developers to think of pages or sites in a more holistic way. By thinking less about the individual elements on a page than the overall look and effect, developers get a better overall view of how their site looks (and by extension works) for others. CSS requires that a developer see a page more like a user; in fact, part of the purpose of CSS is to merge these views and preferences: "it took into account that on the Web the style of a document couldn't be designed by either the author or the reader on their own, but that their wishes had to be combined, or 'cascaded,' in some way" (Lie & Bos, 1999).

This holistic approach can indirectly produce two other beneficial outcomes: a consistent user interface and site-level design patterns. "Redefining a large set of tags to automate text formatting or extensive use of CSS IDs to define a layout can have the

[3] This is the ideal situation, though with varying workarounds and "hacks" necessary to manage the differences in CSS implementation across browsers, HTML data can sometimes end up in strange order without the CSS overlay.

same effect of creating a site-wide design template. Simply changing the style sheet will update the major design elements of any page linked to the CSS file" (Reed & Davies, 2006).

External CSS documents can easily be applied across every page of a site very quickly, tying the user experience together with consistent colors, fonts, and positioning. Any changes to the CSS documents will apply instantaneously to hundreds or thousands of pages. HTML itself cannot accommodate such control. In turn the knowledge of such control feeds into a big-picture approach, in which a developer looks for and creates general methods for tackling possibly thorny problems. In CSS these "design patterns" again encourage developers to think of the site as a whole, feeding the development cycle for better approaches to site-wide, user-friendly style choices.

## *The drawbacks of CSS*

Though CSS was proposed with many positive intentions and goals, the reality is more complicated. Though standards have been published by the W3C, no one Web browser implements CSS directly to the specification. The reason is two-fold and understandable, built directly from the natural tension between concept and implementation. First, since their inception, there has always been fierce competition among the browsers (e.g., the "Browser Wars" of the late 1990s). Such competition makes it desirable for some to ignore standards whether they are too difficult to implement or too costly for tight delivery deadline; or they might not wish to compromise a proprietary, "better" solution. Secondly, these browser developers (and the community at large to some extent) look at Web browsers as "test-beds" for the next generation of Web activity. One need look no further than the highly popular Firefox

Web browser, which was built from the ground up to be the "next generation" browser. And while this browser is more highly standards compliant than most, it does include additional CSS rendering tags that are strictly Firefox-specific (e.g., -moz-corner-radius to allow curved boxes).

*The complications of hierarchy*. Inheritance does make most CSS modifications easy, if thought of from a top-down approach. But after establishing a top-level set of CSS rules, how does one then break those rules? Generally this must be done through new class and identifier specifications for the appropriate subsections of a page or site. This complicates the CSS code by adding a level of verbosity to the rules, including "element#identifier.class.class" style specifications.[4] Not only does this add byte-wise bulk to the CSS document for every time a rule needs to be "broken" but is also complicates the CSS for the designer if it is less than intuitive due to implementation limitations.

*Document management.* As a direct result of the flexibility of CSS hierarchy, as a Web site becomes complicated, its CSS too becomes complicated. Whether it is CSS spread out in documents, style tags or style attributes or a myriad of CSS rules in a single document or a series of CSS documents to be applied in particular contexts, file management quickly becomes an issue. In order to know what files affect what pages in a complex web site, a systematic approach is required—and the CSS community has yet to offer any guidelines on this particular matter.

[4] To further complicate the problem some popular browsers (Microsoft Internet Explorer) do not recognize .class.class structure properly. The result is the need to rely on identifiers, which must be applied one at a time because identifiers are intended to be unique.

Developers tend to either allow a third-party system to "manage" their CSS (e.g. Adobe/Macromedia Dreamweaver, Microsoft FrontPage, or other less commercial content management solutions[5]) or they come up each with her own approach. And often when a developer overcompensates, attempting to make sure the code is there and available in a rushed time frame, redundancy and bloat arise. Moreover, as bloat grows, it becomes hard for multiple developers to know which code is actively in use and which should be deprecated. Managing this CSS in a multi-page, multi-developer environment can quickly become a full time job.

*Best practices.* While it has largely been the tradition within Web development to avoid any enforced systematic approach (e.g., HTML file linking flexibility), more systems are pushing developers to initiate better file control and separation of data types (e.g., XML Cocoon, Java Servers, etc). A similar approach to CSS document management might alleviate some of the aforementioned complications.

If the approach is not strictly enforced, however, the creation of a set of "best practices" standards that address these issues would represent a good compromise. How many stylesheets should a 10-page Web site use? 100 pages? 1000 pages? How should this CSS be implemented? All in one document or top-level styles with subpage styles? In what situations should classes be used instead of identifiers, and vice versa? None of these questions have definitive answers, at least not without context, so a set of guidelines could be a major benefit to novice and expert CSS designers.

[5] Dudek & Wieczorek, 2003.

### *The need for a tool*

Given the enormous utility of CSS, as well as its growing ubiquity in Web development, minimizing its limitations and moving towards best practices standards are both legitimate goals. At a minimum, identifying redundancies and hierarchical simplifications would make a good start. Such efficiency modification has been attempted with Web documents in general (Ricca et al, 2001), but not specifically with CSS.

This investigation will propose and create a tool that will examine a Web site, parse all of its CSS, look for usage patterns, and finally propose aggregations where appropriate. In its initial development, this tool will simply point developers to places where the code can be better written or better managed.[6] It will follow some basic rules about how it gets its CSS then how it compares individual snippets of CSS. These rules will be written and coded as extensibly as possible, so that they can be modified and enhanced in future phases of the project. The tool's pneumonic title is the "CSS Spa," which stands for "Cascading Style Sheet System for Parsing and Aggregation."

## The CSS Spa

Conceptually the CSS Spa is a repository of CSS attribute-value (AV) pairs that are relationally connected to their original sites, pages, styles, classes, and identifiers (see Figure 1). This structure allows the core AV CSS to be examined for commonalities and differences then manipulated appropriately. This examination

[6] Ideally later versions of the tool will rewrite the CSS documents and allow developers to deploy these themselves or have the tool automatically do so. The deployment issue is a further complication that is beyond the scope of this project.

follows a series of basic rules that produce CSS guidelines for the individual page or site (encapsulated in Table 1).

**Table 1. Rules for the CSS Spa.**

| Rule | Purpose |
|---|---|
| 1 | Enforce atomic separation. |
| 2 | Eliminate non-CSS style elements. |
| 3 | Eliminate inline styles. |
| 4 | Normalize attributes and values. |
| 5 | Create new CSS "superclasses." |
| 6 | Remove unused CSS. |

*Rule #1: Enforce atomic separation.* All CSS AV pairs, classes, and identifiers should be located and separated. These include those pairs that exist on the page itself (style tags and attributes) as well as those inherited from parent stylesheets. An immediate benefit of this atomic separation of AV pairs is the natural recognition of unused CSS: if the AV pair is used in the external stylesheet but not referenced on the page itself, it is actually not in use and should be marked for possible removal.

*Rule #2: Eliminate non-CSS style.* The tool's second rule is to examine the now-parsed HTML code for any older HTML tags and try to remove them or at least recommend that they be handled consistently. An example might be the <B> or <STRONG> tags. The recommendation might be to create a SPAN.HEAVY class that could handle all of these instances. The downside of creating such a class is that it is much more verbose (and therefore more bandwidth-unfriendly) than a simple

<STRONG></STRONG> tag set, so an alternative might be to guaranteed that all <B/> and <STRONG/> tags are consolidated into a single tag.[7] Then that tag can be actively styled as a semantic HTML element via the site's CSS documents (e.g., STRONG { font-weight: bold; color: red; }).

This rule should also apply to any standard attribute that can be applied by either CSS or HTML (e.g. width). HTML tables in particular often have border and width attributes that, because they are not semantically important to rendering the table's data, should be managed outside of the HTML if possible.

*Rule #3: Eliminate inline styles.* All inline styles (i.e. CSS styles applied via the STYLE attribute) should be eliminated and replaced (temporarily) with class-based CSS. This will also make later comparison of CSS styles more consistent, but more importantly it will remove all stylistic information from the page itself—following the underlying mantra of separating content from presentation. Inline styles limit the presentation outputs an HTML page can produce because they override all other CSS rules. Removing them allows more flexible output as well as cleaner CSS management, requiring developers only to look in CSS documents, not also HTML files.

*Rule #4: Normalize attributes and values.* After these atomic separations are complete, the CSS should then be rebuilt into common groupings, and simplified where possible. The AV pairs should be examined for common or highly similar attributes (e.g., color: black versus color: #000000) and consolidated. This method of CSS normalization will allow the later rules to be applied more effectively.

[7] <STRONG> is preferable HTML to <B> because it is more semantic and less stylistic. See http://developer.mozilla.org/en/docs/HTML:Element:strong.

*Rule #5: Create new CSS "superclasses."* After the data has been massaged and normalized, examine it for common patterns. If CSS classes or elements share 80% (or higher) similar stylistic content, flag these groups as possible aggregation matches. The recommendation in these cases is that the CSS developer create a "superclass" that covers the 80% behavior, then the other CSS classes can individually express the remaining styles. This rule is focused on efficiency and bandwidth but also will reinforce the cascading nature of CSS and hopefully make developers more aware of their own design patterns for future development.

Any elements, classes, or identifiers that span multiple pages will be aggregated into the site-wide stylesheet. As a site's page-measurable size grows, however, a multi-tiered hierarchy of site-wide and page-grouped CSS documents will be preferable. For its current incarnation, the CSS Spa will largely ignore the site size and simply produce recommendations for either top-level or page-level CSS.

*Rule #6: Remove unused CSS.* A beneficial side-effect of this grouping activity is that any CSS that is not in use will be found. If an element, class, or identifier rule exists *for* a page but is connected to no elements actually *on* the page, it is inactive and should be marked for removal.

## Building the Tool

*Platform.* The CSS Spa has been developed on a MySQL database with an Apache 1.33 and PHP front-end interface. This selection was made largely for the convenience of development (i.e. high familiarity with both systems), but also for the convenience of a later user. Deployment of such an environment is almost trivial, particularly with the help of many prepackaged “AMP” (Apache/MySQL/PHP) distributions available.

*Data structure and storage.* The CSS Spa’s data structure consists of a simple relational model between sites, pages, elements on those pages, and related stylistic information (see Figure 1). As an individual page or site is spidered, its document object model (DOM) is parsed for any possible style information. When found this information is added to the **style** table (if it is not there already) then associated with the proper locations where it was found. Once an entire page or site has been spidered, there will be numerous style elements, page elements, and classes stored in the database for examination.

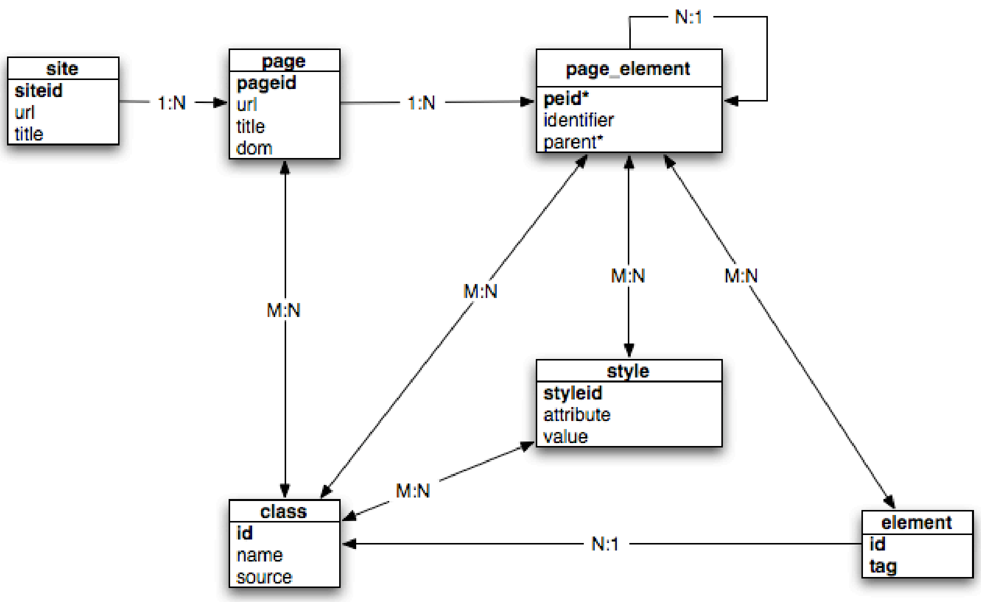
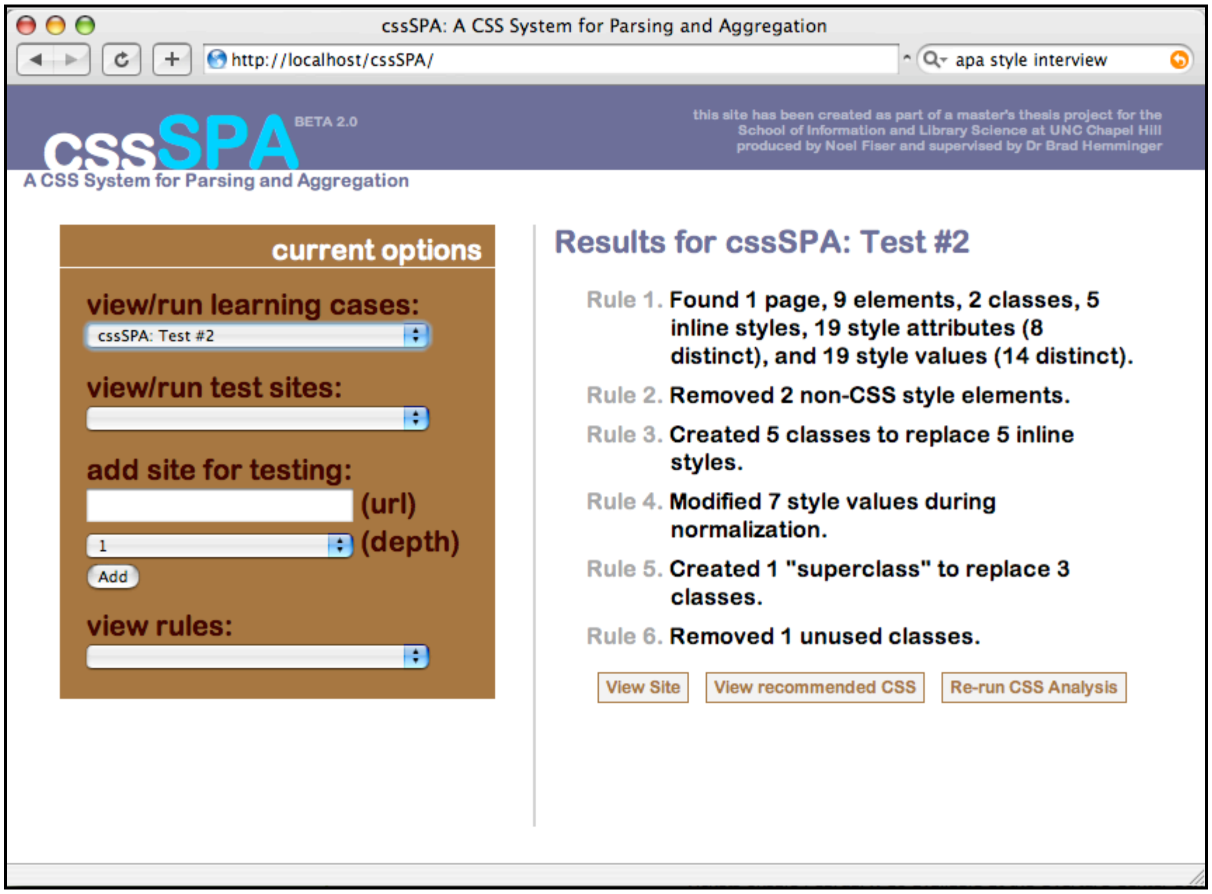


*Figure 1. The CSS Spa entity-relationship diagram.*

*User interface.* The user interface for the CSS Spa intended to be as simple as possible (Figure 2). Because the system itself runs using PHP on an Apache Web server, the user interface itself was also conceived as Web-based. There is only one page to the interface, however, allowing the CSS Spa to act as a dashboard for CSS testing. When a user accesses the CSS Spa Web page, she is presented with a menu of possible activities on the left side of the screen: view learning cases; view test sites; add test sites; and view rules. There are currently three learning cases and three test sites (see below). To add a test site, a user need only enter the URL and select a depth (i.e. how many levels deep on the site should the tool visit). Each rule has a brief description that is accessible from this menu.



this site has been created as part of a master's thesis project for the School of Information and Library Science at UNC Chapel Hill produced by Noel Fiser and supervised by Dr Brad Hemminger

Results for cssSPA: Test #2

Rule 1. Found 1 page, 9 elements, 2 classes, 5 inline styles, 19 style attributes (8 distinct), and 19 style values (14 distinct).

Rule 2. Removed 2 non-CSS style elements.

Rule 3. Created 5 classes to replace 5 inline styles.

Rule 4. Modified 7 style values during normalization.

Rule 5. Created 1 "superclass" to replace 3 classes.

Rule 6. Removed 1 unused classes.

*Figure 2.* *The CSS Spa user interface.*

When a user selects an option, an HTML iframe on the right side of the page loads the corresponding information from the database and displays it. When a user selects a learning case or test site, the iframe displays the summary results for the site's last run through the CSS Spa tool. A "more details" button allows the user to open up a new window with a more detailed report on the case or site, including the summary information, all page URLs involved, and its current CSS Spa stylesheet. Another button in the iframe labeled "run again" will allow the user to rerun the CSS Spa rules over the site. When this option is selected (or when a new site is added in the left menu), the iframe reloads with activity information as each of the rules is applied.

## *Learning Cases and Test Sites*

Three learning cases were developed in order to test the CSS Spa. After these learning cases the tool would then be applied to three different but internally known Web sites to see what recommendations the CSS Spa would produce for real-world data.

The first learning case is a page with all of its CSS style information placed inline using the STYLE attribute on various HTML elements. This learning case is intended to hone Rules 1-3 above. The second case is a similar Web page with similar classes and identifiers that will practice Rules 4-6, with particular emphasis on #5 ("Create new CSS groups"). The final learning case is a multi-page site that will be analyzed in order to create a single uniform stylesheet. This will also hone rule #5.[8]

[8] All test pages are available in Appendix A.

*Site #1: Low CSS.* The three test sites are various projects the tool's designer has created over the last 10 years, at varying stages of CSS usage and development. The first is the Horizon site, http://horizon.unc.edu/. This site was created in its current form in 1997 using very little if any CSS. What CSS it contains has been added since and makes up only a small percentage of the site. The Horizon site is highly organic and very large (1000's of different pages, though often in a single template). Its development has only been rudimentarily controlled by a header/footer template built around HTML edited by non-experts using Microsoft FrontPage editor. As a result, the pages exhibit wide stylistic and structural variation.

*Site #2: High CSS, mostly inline.* The second site being tested is the online journal *Innovate*, http://innovateonline.info/. It is of moderate size (20 to 30 page types), as most of its page structure is built based on consistent data sources from a common database (e.g. issues, articles, biographies, etc). It was created in 2003 and almost exclusively uses inline CSS, though it also has a site-wide stylesheet. The major benefit of the CSS Spa for this site would be to streamline all of its CSS into top-level stylesheets.

*Site #3: High CSS, well-structured.* The final site tested by the CSS Spa is the *Technology Source* Archives, http://technologysource.org/. The *Technology Source* was an online journal started in 1998 by Microsoft and made defunct in 2005 after changing ownership a few times. While the journal no longer exists, the articles themselves were free to be distributed with only simple origination attribution necessary. So a new site with a new look was developed to house the original *Technology Source* articles. Because this new journal site was functionally even simpler than *Innovate*, and because of prior experience building that Web site, the *Technology Source* Archives site is stylistically completely controlled by a master stylesheet. As a result, the CSS Spa's recommendations for this site should be minimal.

## Results

### *Learning Cases*

When the first CSS Spa rule (atomic separation) is invoked, the target site is systematically crawled, each page being captured and its DOM being parsed for elements, classes, identifiers, and style attributes. As each of these is found, it is inserted into the relational database. The results of applying this rule to the learning cases are encapsulated in Table 2 (below). The important structural differences in the pages are reflected in these results: Test #1 and Test #2 are largely the same XHTML/CSS pages, but Test #2 has only inline styles in its code. Its number of classes is indeed zero, but its inline styles are increased to seven to match the total classes and inline styles of Test #1. This is intentional in order that their CSS Spa results can be compared. Also, Test #3 is largely the same page as Tests #1 and #2; it has just been repeated three times and these pages then had hypertext intra-linking added to them. As a result, the number of elements has increased by roughly three times, as well as the counted AV pairs.

*Table 2. Raw results for learning cases.*

| | Test #1 | Test #2 | Test #3 |
|---|---|---|---|
| Pages | 1 | 1 | 3 |
| Elements | 9 | 9 | 33 |

| | | | |
|---|---|---|---|
| Classes | 2 | 0 | 3 |
| Styles (Inline) | 5 | 7 | 15 |
| Attribute-Value Pairs | 19 | 19 | 45 |

Figure 3 (below) offers a visual representation of these results. The values are presented as percentages of the total values found. For example, among the learning cases there were a total of five pages: one for the first two (20% each); and three for Test #3 (60%). This percentage-based scale allows these sometimes-disparate categories to be displayed side-by-side for clear comparison.

Notable in the learning cases is how the complexity of the process increases dramatically when multiple pages are added (Test #3). In each case, Test #3's values are additive to those of the original page, so as the number of pages in a site increases, so do all the factors that influence CSS application in a direct relation to the total number of pages on the site. This observation will become increasingly obvious and important when applied to real-world sites (e.g. the test sites to follow), where the number of pages is always dramatically larger than these learning cases.

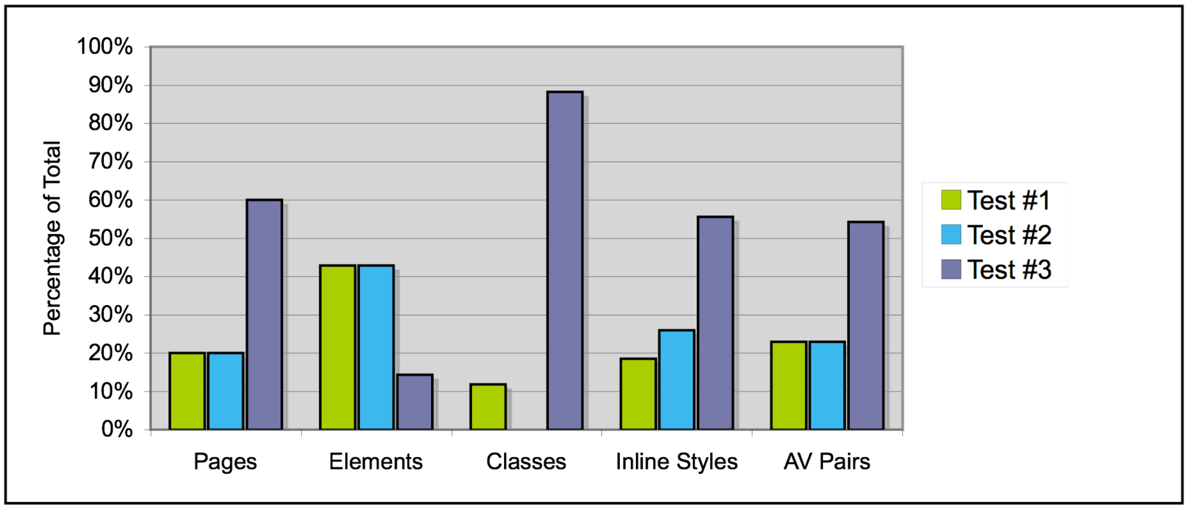


*Figure 3. Graph of results for learning cases (as percentages of total found).*

The results of applying CSS Spa rules 2-6 to the learning cases are encapsulated in Table 3 (below). Generally the results are consistent with an inherent relationship between the corresponding values. For rule #2 ("Remove non-CSS styles"), for example, the number of elements removed corresponds to the number of pages involved as well as the number of non-CSS style elements per page. Interestingly, rule #5 shows an explosion of styles being added to the superclasses; in part, this has to do with the normalization of CSS attributes (e.g., "border" into 12 atomic attributes: "border-color-top," border-color-left," etc.), but also of course relates again to pages involved in the analysis. Also, as superclasses become more prevalent, the number of unused classes also increases and can be excised (rule #6).

*Table 3. Rule results for learning cases. () indicates rule number.*

| | Test #1 | Test #2 | Test #3 |
|---|---|---|---|
| Non-CSS Styles (2) | 2 | 2 | 16 |
| Inline Styles (3) | 5 | 9 | 15 |
| Normalization (4) | 7 | 4 | 6 |
| Superclasses (5) | 1 | 1 | 4 |
| Styles in superclasses (5) | 2 | 9 | 68 |
| Unused CSS (6) | 0 | 0 | 2 |

## ***Test Sites***

Having honed the CSS Spa functionality using the learning cases, it's ready to apply to the three test sites.[9] The number of structural units in each site varies widely but intra-relationally (e.g. the site's elements per page or attributes per style ratios are within the same order of magnitude). Noticeably the *Technology Source Archives* (*TSA*) has 1343 of its pages accessible to the CSS Spa, resulting in a much larger set of resultant values. *Innovate* on the other hand has only 20 pages, though it contains many more elements per page than the *TSA*.

The number of CSS classes and styles in the *Horizon Site* structure is striking, though less so when considering its development prior to the adoption of CSS by the major browsers.[10] Conversely, the number of CSS classes and styles in the other two (CSS-laden) sites grows rapidly in relation to the number of pages—though not as

[9] Each test site was tested to a depth level of 3, meaning every link was crawled in a pyramid fashion from the first page to its children pages and finally their children pages. Unlike the other two, for the third test site, the *Technology Source Archives*, this depth actually covered every page on the site.
[10] As a corollary, this is probably also why its ratio of elements to pages remains very high, since styling was applied through HTML elements instead of CSS classes.

quickly as the number of elements does. One might hypothesize that the AV pairs are distributed between the CSS classes and inline styles, and thus do not grow as rapidly as the number of elements per page naturally must.

*Table 4. Raw results for test sites.*

| | Horizon Site | Innovate | Technology Source |
|---|---|---|---|
| Pages | 164 | 20 | 1343 |
| Elements | 14228 | 4529 | 98189 |
| Classes | 11 | 133 | 3104 |
| Styles (Inline) | 168 | 1636 | 9653 |
| Attribute-Value Pairs | 443 | 3890 | 21433 |

Figure 4 (below) displays the relationships between the three sites. Worth noting is the noticeably different natures of the sites. While the most obvious difference lies in the massive number of pages (and therefore structural elements) in the *TSA*, it is also worth noting the inverse relationship between the non-CSS site (*Horizon*) and the two CSS sites. The majority of structural elements evident in the *Horizon Site* are pages and elements (columns 1 and 2 in Figure 4), whereas these are very much smaller (or at least equal) in the CSS-based sites, with increasing emphasis naturally on CSS structures (columns 3-5).

Also, comparing columns 3 and 4 reveals an inherent difference between two of the sites: implementation of classes (*TSA*) more heavily than inline styles (*Innovate*). This was a documented feature of the test sites, so it is reinforcing to see a consistent result from the CSS Spa's output.

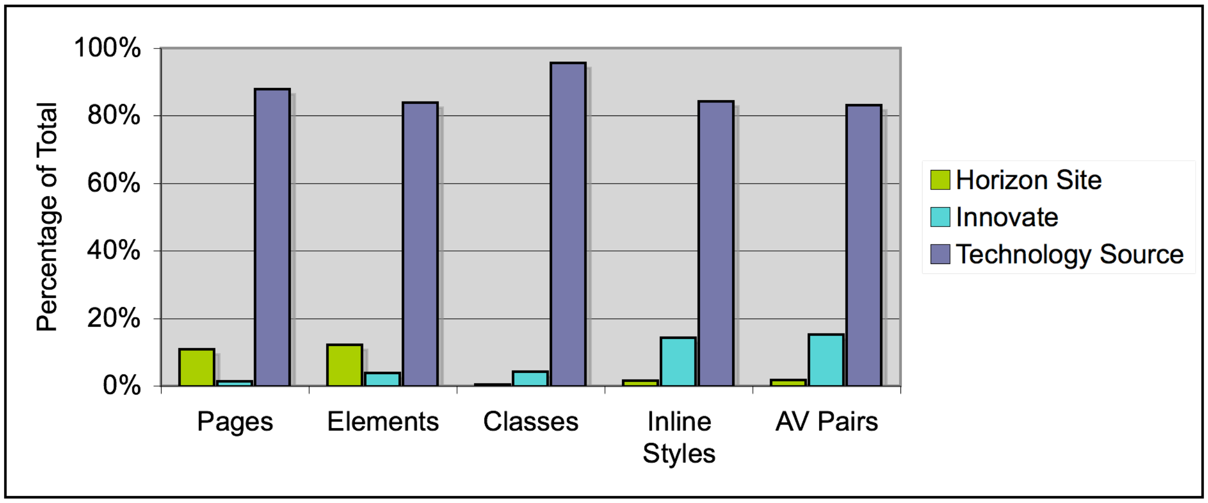


*Figure 4. Graph of results for test sites (as percentages of total found).*

As rules 2-6 were applied to the test sites, patterns similar to those observed in the learning cases emerged. The same explosion of styles in superclasses is observable in rule #5 here as well as the correlated increase in how much unused CSS can be removed (rule #6). Interestingly, the normalization of style values is only as effective as the site designer(s) themselves. That is, it is not directly correlated to any other chronological or technological factor (unlike the non-CSS styles removal rule, which is directly affected by what version of HTML the site was written for). If the designer has been consistent with color and other style choices, the normalization will naturally find fewer "abnormal" styles to correct. In the case of the *TSA*, which has many, many more pages than *Innovate*, for example, there are roughly half the normalizations, indicating a more selective and controlled use of CSS.

*Table 5. Rule results for test sites. () indicates rule number.*

| | Horizon Site | Innovate | Technology Source |
|---|---|---|---|
| Non-CSS Styles (2) | 2367 | 222 | 9671 |
| Inline Styles (3) | 168 | 1636 | 9653 |
| Normalization (4) | 6 | 73 | 41 |
| Superclasses (5) | 15 | 50 | 9 |
| Styles in superclasses (5) | 9381 | 393824 | 4550 |
| Unused CSS (6) | 182 | 91 | 7585 |

## Analysis

By examining the results of the learning cases and test sites, the utility of the CSS Spa tool utility can be ascertained more fully. It must be noted, however, that the test site results are certainly more random and harder to correlate because they do not exist isolated in a vacuum; that is, they are not tightly controlled like the learning cases.

The patterns that emerge from three sites do offer some limited insight into real-world Web design but must be labeled as theories at best.

But before examining the results of running the more complicated rules, it is enlightening to dissect the initial results of rule #1 (atomic separation), looking for any noticeable differences between the two test groups. Figures Figure ***5*** and Figure ***6*** (below) illustrate some basic similarities and differences between the sites.

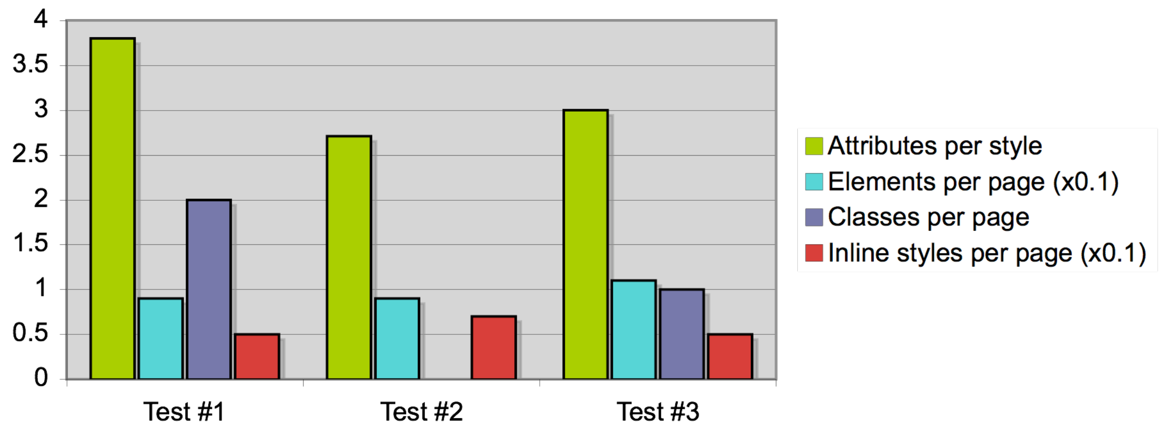


*Figure 5. Per-page analysis of learning cases.*

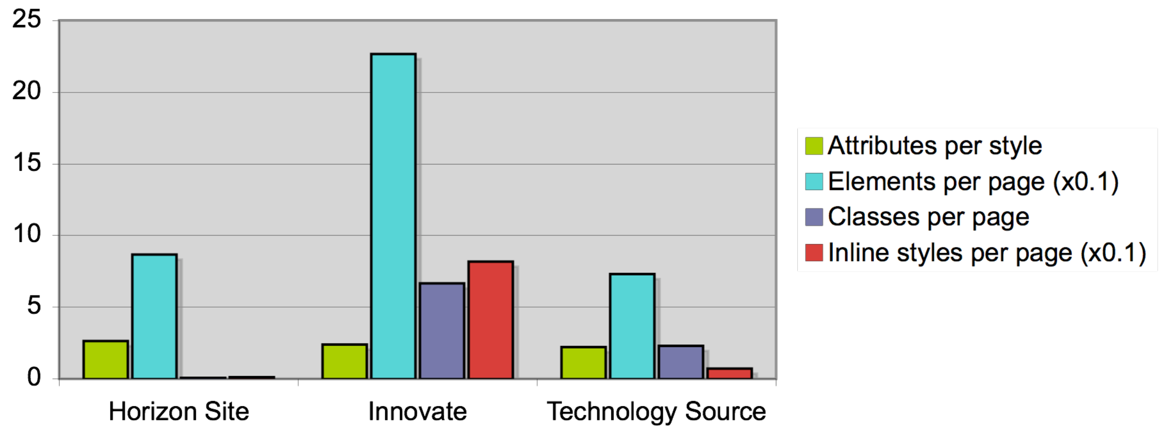


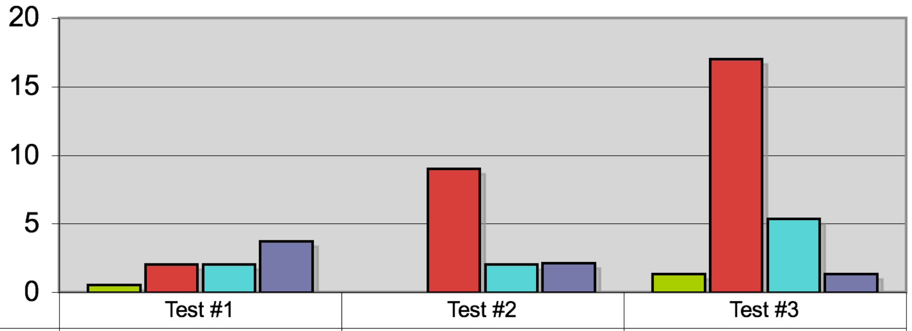
*Figure 6. Per-page analysis of test sites.*

The learning cases present a slightly higher number of attributes per style than do the real-world sites, as well as considerably fewer elements per page than most any site employing even the most basic XHTML and CSS. Also worth noting is the decrease in classes per page (column 3) in the real-world examples. It is almost non-existent on the non-CSS *Horizon Site* due to its massive number of pages. Ignoring that site for the moment, it is important to note that a possible pattern has emerged in these results: even though the *TSA* site is more class-oriented than *Innovate*, its overall classes per page (CPP) figure is lower. One could speculate that this is the result of a more consistent use of classes across multiple pages (i.e. through master stylesheets). A more uniform design utilizing standardized CSS would naturally allow more pages to be added with fewer additional classes and styles to be added, thus decreasing the CPP even further.[11] It is equally valid, however, to note the functional differences between

[11] It is somewhat worth noting that this behavior is independent of the ostensibly related figure for "inline styles per page." The more "inline" *Innovate* site does indeed have a higher value here, whereas the *TSA*'s remains very low.

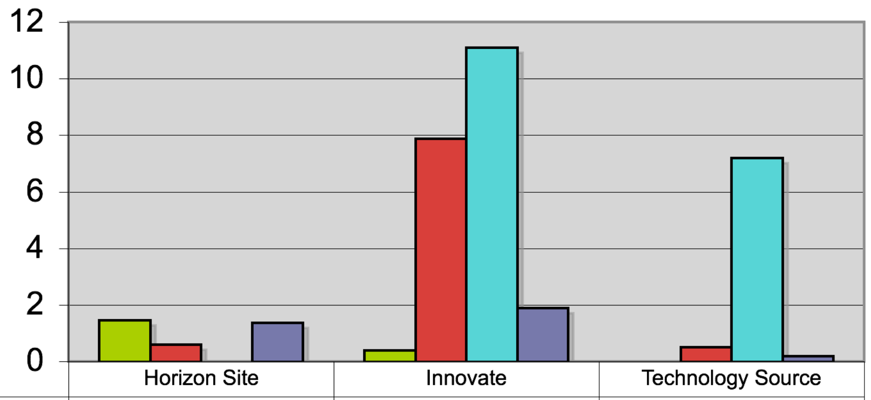
these two sites: the *Innovate* site has many more interactive elements (discussion boards, email-a-friend options, and even membership requirements) and thus has more different types of pages to render. Its CPP figure is likely to rise as a direct result.

*Class merging.* Among the learning cases, class merging was numerically effective only in Test #1, where one superclass was created for the information in two original classes (see Figures Figure 7 Figure 8 below). In Test #2, where there were no classes to start, one superclass was formed. And in Test #3, more superclasses were formed than there were classes to begin with. Clearly, the raw efficiency of class merging with a small number of elements, classes, and styles is situational at best, poor at worst.



| | Test #1 | Test #2 | Test #3 |
|---|---|---|---|
| 🟩 Change in Classes | 0.5 | 0 | 1.33 |
| 🟥 Superclass Efficiency | 2 | 9 | 17 |
| 🟦 Non-CSS Removed (per page) | 2 | 2 | 5.33 |
| 🟪 Normalizations (per 10 A-V) | 3.68 | 2.11 | 1.33 |

*Figure 7. Rule analysis for learning cases.*



| | Horizon Site | Innovate | Technology Source |
|---|---|---|---|
| 🟩 Change in Classes | 1.45 | 0.38 | 0 |
| 🟥 Superclass Efficiency (per 1000 A-V) | 0.5863 | 7.87648 | 0.505555556 |
| 🟦 Non-CSS Removed (per page) | 0.01 | 11.1 | 7.2 |
| 🟪 Normalizations (per 100 A-V) | 1.35 | 1.88 | 0.19 |

*Figure 8. Rule analysis for test sites.*

On the other hand, when the number of elements on the page increases dramatically, as is the case for all three real-world test sites, the efficiency of the CSS Spa begins to surface. In particular, the number of superclasses produced on the *Innovate* site (50) represents a significant reduction from the original amount (133). In the case of the *Horizon Site*, however, where there was very limited CSS applied to begin with, the efficiency was again reduced (11 originals become 16 superclasses). These 16 superclasses include the information from 11 original classes and 9,381 classes created from inline CSS, so the overall utility of these superclasses more than compensates for it.

*Superclass efficiency.* The true measure of efficiency in the CSS Spa is not how many classes are produced but how many normal classes as well as "inline classes" (from rule #3) can be subsumed under the superclasses. In other words, if a site does end up with more classes but many of those classes contain information that covers many other styles, the creation of the class is worth it. This is particularly important when enforcing normalized (and therefore atomized) styles as well as when substituting classes for inline styles; each of these processes can create many more individual classes with specific style information and it is the purpose of the superclass to encapsulate these.

Among the learning cases, superclass efficiency increases dramatically as more inline styles and more pages are introduced. In Test #2, the information from nine individual styles gets included in a single class; and in Test #3, 68 common attributes are merged into 4 superclasses. So even on a smaller scale the efficiency of creating these common-attribute classes can save design time and ultimately visitor bandwidth.

It is in the real-world examples, however, where this superclass efficiency becomes most manifest. As already alluded to, the *Horizon Site* encompasses the highly repetitious information of 9,381 attribute-value pairs within just 16 superclasses. If this isn't impressive enough, over just the 20 pages of the *Innovate* site, inline and non-CSS information combines with the original 133 classes to create a massive number of new class values. Impressively, however, 393,824 of these attribute-value pairs can be encompassed in just 50 new superclasses.

The particular efficiency of superclass creation in all of these cases points to two things: though there may be liberal use of inline styling, these sites do reveal a general style consistency on the part of the designer. This is no surprise, since the look and feel of a Web site should remain generally consistent throughout, no matter how many pages are involved. But secondarily this efficiency also points to a limitation in the point-of-view of the CSS Spa tool: it's running against the XHTML and CSS as rendered in the browser, so even if Web pages may be delivered from a small number of templates on the server side, the browser sees each page individually. There is no terrible harm in this, as the CSS created by the tool will be just as consistent and as efficient as the original templates are. But the client-side point-of-view does lend to the overall magnitude involved in spidering these sites. As a result, it might be more efficient to select 10-20 example pages for the CSS Spa to analyze instead of naturally flowing through the tiers of the given site.

*Non-CSS styling per page.* Another efficiency to watch (and one that certainly contributes to the superclass efficiency as well) is the number of non-CSS style elements that are removed from each page on the site. Identifying these non-CSS style elements like <B/> and <I/> tags is more useful for normalization than it is for replacement; that is, if all <STRONG/> and <B/> tags can be merged to <B/> then the styling on these tags can be managed in a single statement for the sake of consistency.[12]

In the learning cases Tests #1 and #2 each had two style elements converted. Over the three pages of Test #3, 16 style elements were transformed—a much higher ratio (5.33). On the real sites, these ratios varied more widely. Surprisingly, the largely non-CSS *Horizon Site* contained only one such element over its 164 pages. So though the site uses little CSS, what it does use it does so efficiently. On both *Innovate* and the *TSA*, even though one site had dramatically fewer pages (20 versus 1343, respectively), the usage (and replacement) of these style elements was similar: 11.1 and 7.2 elements per page, respectively. Thus, just like efficient master stylesheets allow many pages to

[12] On the other hand, conversion of <FONT/> tags to CSS font-family and font-size attributes is much more desirable in practical terms: the <FONT/> tag has been deprecated in the HTML specifications—though most modern browsers still render it properly.

be delivered with similar styling, the appropriate and consistent use of style elements can lead to similar efficiency across 10, 100, or 1000 pages of a site.

*Normalizations.* Attribute normalization is another method of controlling CSS sprawl. In the CSS Spa it was applied largely to five CSS attributes: border, background, color, margin, and padding. Standardization of margin and padding, for example, allows a style to be applied in as few CSS statements as possible. Similarly, normalization of color names and RGB values into hexadecimal values guarantees that all similar color sets can be related.

In the Test #1 among the learning cases, 19 attribute-value pairs underwent 7 normalizations (or 36%) for the aforementioned CSS attributes. 21% were normalized in Test #2 and 13% in Test #3.

For the test sites, normalization affects only a small percentage of attribute-value pairs. The *Horizon Site* had just 6 normalized values for its 443 attributes (1.3%). *Innovate* had 1.9% normalizations (73 among 3,890 attribute-value pairs). And the *TSA* had only 41 normalizations in 21,433 pairs (0.02%). It is important to keep in mind, however, that these normalizations occur on a single attribute-value pair that could be repeated many times over the many styles and classes associated with a site. And clearly, because the normalization has a very beneficial effect in aiding superclass creation and efficiency, even by making sure that every reference to the color “black” or “rgb(0,0,0)” is converted to “#000000,” for example, the designer increases by three times his chances of seeing and correcting varying CSS coding habits. The normalization process encourages and ultimately enforces best practices behavior, or at least a design pattern scheme.

## Future Recommendations

Based on the limited testing thus far, the CSS Spa has proven itself to be a useful tool for site designers wishing to complete a site-wide analysis for streamlining their CSS code. This does not mean, however, that it cannot be improved in many, varying ways. For example, just adding more real-world sites for testing should reveal more CSS usage patterns, those blatant as well as more subtle.

*Speed.* In order to equip the CSS Spa for such testing, however, it would probably be best to reconstruct it in a more efficient (i.e. compiled) programming language like Java. As it is, though the CSS Spa has been rewritten twice to increase its efficiency and speed at accessing sites, it remains inordinately slow at crawling large sites (e.g., the *TSA* can take up to hours to crawl). Site crawling is also currently monolithic, so when there is a network glitch or a server stops responding, the crawl must be restarted. Similarly, the rules can be run individually, but each one is monolithic in itself, meaning that when it fails to complete, it must be run once again from the beginning.

A simpler modification that might enhance the CSS Spa would be the addition of CSS state handlers for elements and classes. The CSS Spa is aware of the differences between classes and identifiers but currently ignores state (e.g., “hover”) because it is not a fully supported feature across the major browsers. Still, it is common for designers to invoke this property when creating site CSS, so adding it to the CSS Spa repertoire would be useful and appropriate.[13]

[13] Testing for states like “hover” would also allow the CSS Spa to flag them as non-standard when they are applied that way. For example, Mozilla Firefox allows the application of state to any element while Microsoft Internet Explorer (MSIE) only

*Customization.* Beyond functional limitations of the CSS Spa, it is equally important that the tool be flexible enough to allow its users to customize its behaviors. For example, giving the user the ability to tweak how similar classes should be before rule #5 creates a superclass would enable a designer to use the tool to "tighten" or "loosen" a site's CSS according to his wishes. Though it now looks for 80% matches, if a designer can "clean up behind himself" by instead setting this value to 100%, so that only redundant classes would be merged. A similar argument might be made for loosening the rule to allow many more superclasses to be created, particularly when a site has a long history of different designers and design patterns.

Ideally this customization should also allow the user to select particular elements, classes, or styles that he wishes to examine more closely. The natural usage patterns of such structural elements may reveal a "best practices" guide for a site. And in striving for such best practices as these, to allow a designer to enforce custom rules, like font families and color schemes would also be infinitely useful in reinforcing site integrity. Allowing a user to designate an "ideal" stylesheet and have the CSS Spa compare its classes with the actual site, returning a report of discrepancies would save great time and effort.

One more complicated goal of the CSS Spa that had to be deferred due to time constraints is the ability to create and recommend tiered CSS documents, so not only a master stylesheet for a site, but other stylesheets for groups of pages which are similar to one another but not so much to the rest of the site. In some sense this is treating these

allows it on <A/> tags. While Firefox's behavior is standards-based, it would be useful to alert the designer to these state references that will not work in MSIE, still the most common browser on the market.

groupings as “sub-sites” and thus might be relatively easy to implement in the current framework by creating a secondary relationship between sites, these sub-sites, and the pages below them.

## Conclusion

The CSS Spa is an initial attempt to offer Web designers a new tool for examination, aggregation, and standardization of XHTML and CSS. As has been demonstrated, the tool is effective at finding rules-based patterns and redundancies in CSS code and reducing the overall size and complexity of that code. In its current form, however, the tool remains a somewhat primitive proof-of-concept, limited to the ability to crawl a Web site and apply six basic CSS-related rules to it. Future enhancements in customization and efficiency will allow the CSS Spa to be released to a wider audience.

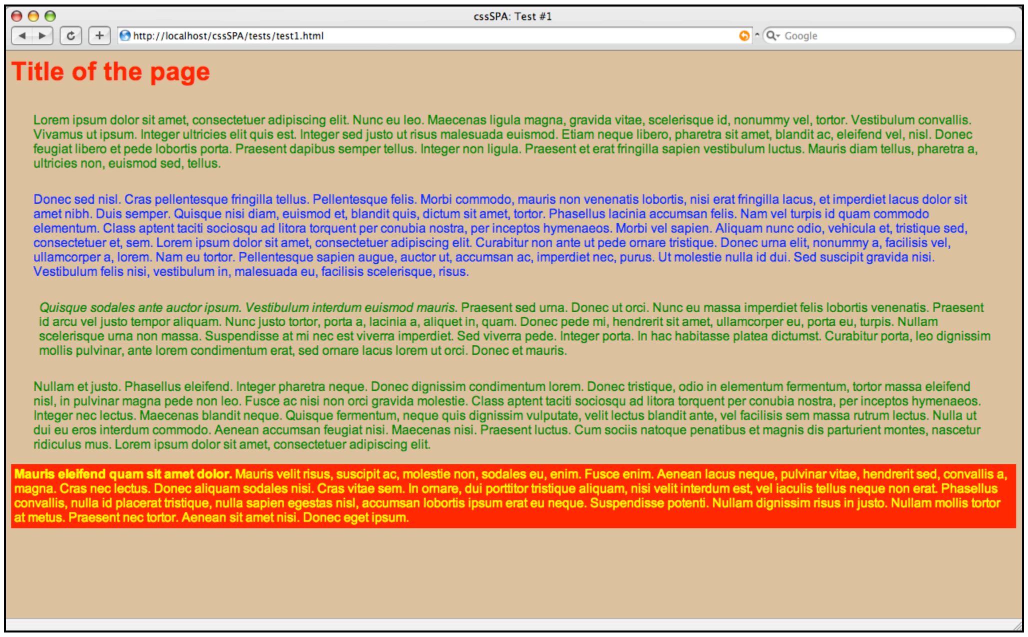
It is clear from the complexity of XHTML and CSS—as well as a fast-growing reliance on the World Wide Web as the world’s most popular communication medium—that a tool to enhance the efficiency and usability of CSS can play a crucial role. Not only can such a tool increase bandwidth efficiencies by reducing redundant code, it can also lead the way to better team-oriented CSS development through its rule-based insistence on best practices, as well as its general encouragement of design patterns.

Over ten years ago, Tim Berners-Lee aptly noted that in the early days of the Web “every browser had its own flavor of HTML. So it was very difficult to know what you could put in a Web page and reliably have most of your readership see it.” (Oakes, 1999). Thankfully HTML, along with its companion CSS, has largely standardized since then, with only a few differences between how each browser renders the same code.  Today the language may be more or less the same but each of so many international developers can implement it in his own Web site (or even page-to-page) in highly idiosyncratic ways. The CSS Spa can allow these sites to undergo a “translation” of sorts, bringing the underlying CSS closer to Web standards as well as bringing the developer closer to best practices.

# Appendix A: Learning Cases

All HTML code related to these learning cases is available online at http://noelfiser.com/cssSPA/tests/.

Test #1 (http://noelfiser.com/cssSPA/tests/test1.html)



Title of the page

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Nunc eu leo. Maecenas ligula magna, gravida vitae, scelerisque id, nonummy vel, tortor. Vestibulum convallis. Vivamus ut ipsum. Integer ultricies elit quis est. Integer sed justo ut risus malesuada euismod. Etiam neque libero, pharetra sit amet, blandit ac, eleifend vel, nisl. Donec feugiat libero et pede lobortis porta. Praesent dapibus semper tellus. Integer non ligula. Praesent et erat fringilla sapien vestibulum luctus. Mauris diam tellus, pharetra a, ultricies non, euismod sed, tellus.

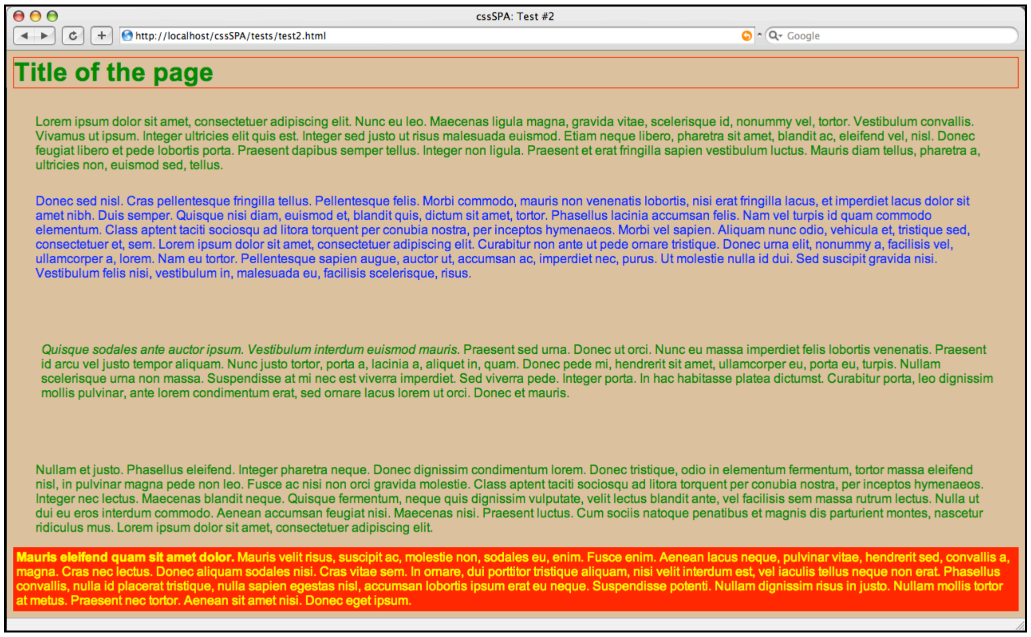
Donec sed nisl. Cras pellentesque fringilla tellus. Pellentesque felis. Morbi commodo, mauris non venenatis lobortis, nisi erat fringilla lacus, et imperdiet lacus dolor sit amet nibh. Duis semper. Quisque nisi diam, euismod et, blandit quis, dictum sit amet, tortor. Phasellus lacinia accumsan felis. Nam vel turpis id quam commodo elementum. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Morbi vel sapien. Aliquam nunc odio, vehicula et, tristique sed, consectetuer et, sem. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Curabitur non ante ut pede ornare tristique. Donec urna elit, nonummy a, facilisis vel, ullamcorper a, lorem. Nam eu tortor. Pellentesque sapien augue, auctor ut, accumsan ac, imperdiet nec, purus. Ut molestie nulla id dui. Sed suscipit gravida nisi. Vestibulum felis nisi, vestibulum in, malesuada eu, facilisis scelerisque, risus.

*Quisque sodales ante auctor ipsum. Vestibulum interdum euismod mauris.* Praesent sed urna. Donec ut orci. Nunc eu massa imperdiet felis lobortis venenatis. Praesent id arcu vel justo tempor aliquam. Nunc justo tortor, porta a, lacinia a, aliquet in, quam. Donec pede mi, hendrerit sit amet, ullamcorper eu, porta eu, turpis. Nullam scelerisque urna non massa. Suspendisse at mi nec est viverra imperdiet. Sed viverra pede. Integer porta. In hac habitasse platea dictumst. Curabitur porta, leo dignissim mollis pulvinar, ante lorem condimentum erat, sed ornare lacus lorem ut orci. Donec et mauris.

Nullam et justo. Phasellus eleifend. Integer pharetra neque. Donec dignissim condimentum lorem. Donec tristique, odio in elementum fermentum, tortor massa eleifend nisl, in pulvinar magna pede non leo. Fusce ac nisi non orci gravida molestie. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Integer nec lectus. Maecenas blandit neque. Quisque fermentum, neque quis dignissim vulputate, velit lectus blandit ante, vel facilisis sem massa rutrum lectus. Nulla ut dui eu eros interdum commodo. Aenean accumsan feugiat nisi. Maecenas nisi. Praesent luctus. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Lorem ipsum dolor sit amet, consectetuer adipiscing elit.

**Mauris eleifend quam sit amet dolor.** Mauris velit risus, suscipit ac, molestie non, sodales eu, enim. Fusce enim. Aenean lacus neque, pulvinar vitae, hendrerit sed, convallis a, magna. Cras nec lectus. Donec aliquam sodales nisi. Cras vitae sem. In ornare, dui porttitor tristique aliquam, nisi velit interdum est, vel iaculis tellus neque non erat. Phasellus convallis, nulla id placerat tristique, nulla sapien egestas nisl, accumsan lobortis ipsum erat eu neque. Suspendisse potenti. Nullam dignissim risus in justo. Nullam mollis tortor at metus. Praesent nec tortor. Aenean sit amet nisi. Donec eget ipsum.

Test #2 (http://noelfiser.com/cssSPA/tests/test2/html)



**Title of the page**

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Nunc eu leo. Maecenas ligula magna, gravida vitae, scelerisque id, nonummy vel, tortor. Vestibulum convallis. Vivamus ut ipsum. Integer ultricies elit quis est. Integer sed justo ut risus malesuada euismod. Etiam neque libero, pharetra sit amet, blandit ac, eleifend vel, nisl. Donec feugiat libero et pede lobortis porta. Praesent dapibus semper tellus. Integer non ligula. Praesent et erat fringilla sapien vestibulum luctus. Mauris diam tellus, pharetra a, ultricies non, euismod sed, tellus.

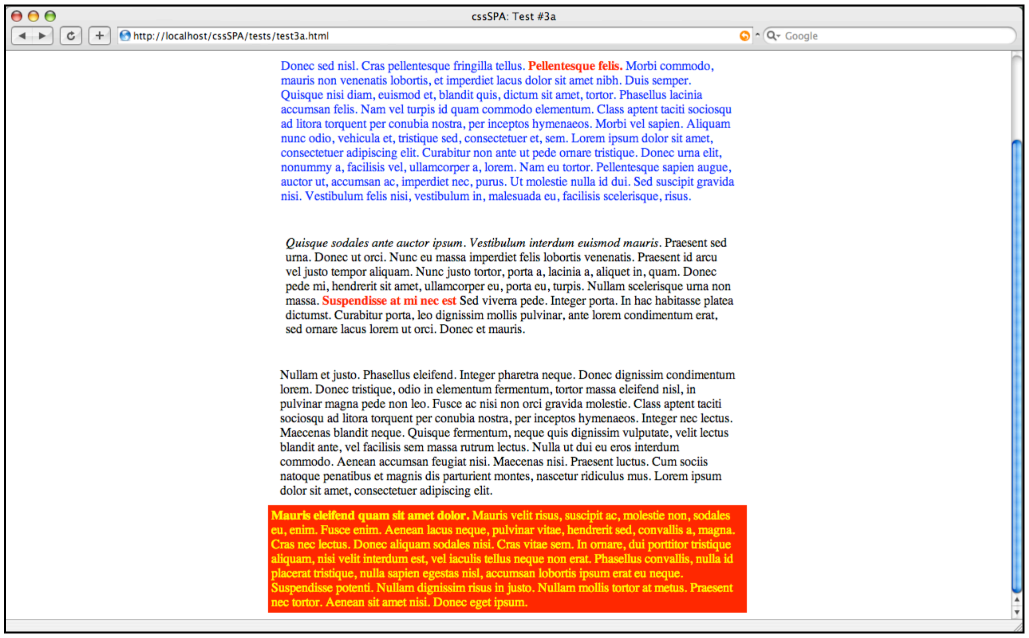
Donec sed nisl. Cras pellentesque fringilla tellus. Pellentesque felis. Morbi commodo, mauris non venenatis lobortis, nisi erat fringilla lacus, et imperdiet lacus dolor sit amet nibh. Duis semper. Quisque nisi diam, euismod et, blandit quis, dictum sit amet, tortor. Phasellus lacinia accumsan felis. Nam vel turpis id quam commodo elementum. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Morbi vel sapien. Aliquam nunc odio, vehicula et, tristique sed, consectetuer et, sem. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Curabitur non ante ut pede ornare tristique. Donec urna elit, nonummy a, facilisis vel, ullamcorper a, lorem. Nam eu tortor. Pellentesque sapien augue, auctor ut, accumsan ac, imperdiet nec, purus. Ut molestie nulla id dui. Sed suscipit gravida nisi. Vestibulum felis nisi, vestibulum in, malesuada eu, facilisis scelerisque, risus.

*Quisque sodales ante auctor ipsum. Vestibulum interdum euismod mauris.* Praesent sed urna. Donec ut orci. Nunc eu massa imperdiet felis lobortis venenatis. Praesent id arcu vel justo tempor aliquam. Nunc justo tortor, porta a, lacinia a, aliquet in, quam. Donec pede mi, hendrerit sit amet, ullamcorper eu, porta eu, turpis. Nullam scelerisque urna non massa. Suspendisse at mi nec est viverra imperdiet. Sed viverra pede. Integer porta. In hac habitasse platea dictumst. Curabitur porta, leo dignissim mollis pulvinar, ante lorem condimentum erat, sed ornare lacus lorem ut orci. Donec et mauris.

Nullam et justo. Phasellus eleifend. Integer pharetra neque. Donec dignissim condimentum lorem. Donec tristique, odio in elementum fermentum, tortor massa eleifend nisl, in pulvinar magna pede non leo. Fusce ac nisi non orci gravida molestie. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Integer nec lectus. Maecenas blandit neque. Quisque fermentum, neque quis dignissim vulputate, velit lectus blandit ante, vel facilisis sem massa rutrum lectus. Nulla ut dui eu eros interdum commodo. Aenean accumsan feugiat nisi. Maecenas nisi. Praesent luctus. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Lorem ipsum dolor sit amet, consectetuer adipiscing elit.

**Mauris eleifend quam sit amet dolor.** Mauris velit risus, suscipit ac, molestie non, sodales eu, enim. Fusce enim. Aenean lacus neque, pulvinar vitae, hendrerit sed, convallis a, magna. Cras nec lectus. Donec aliquam sodales nisi. Cras vitae sem. In ornare, dui porttitor tristique aliquam, nisi velit interdum est, vel iaculis tellus neque non erat. Phasellus convallis, nulla id placerat tristique, nulla sapien egestas nisl, accumsan lobortis ipsum erat eu neque. Suspendisse potenti. Nullam dignissim risus in justo. Nullam mollis tortor at metus. Praesent nec tortor. Aenean sit amet nisi. Donec eget ipsum.

Test #3 (http://noelfiser.com/cssSPA/tests/test3a.html)



Donec sed nisl. Cras pellentesque fringilla tellus. **Pellentesque felis.** Morbi commodo, mauris non venenatis lobortis, et imperdiet lacus dolor sit amet nibh. Duis semper. Quisque nisi diam, euismod et, blandit quis, dictum sit amet, tortor. Phasellus lacinia accumsan felis. Nam vel turpis id quam commodo elementum. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Morbi vel sapien. Aliquam nunc odio, vehicula et, tristique sed, consectetuer et, sem. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Curabitur non ante ut pede ornare tristique. Donec urna elit, nonummy a, facilisis vel, ullamcorper a, lorem. Nam eu tortor. Pellentesque sapien augue, auctor ut, accumsan ac, imperdiet nec, purus. Ut molestie nulla id dui. Sed suscipit gravida nisi. Vestibulum felis nisi, vestibulum in, malesuada eu, facilisis scelerisque, risus.

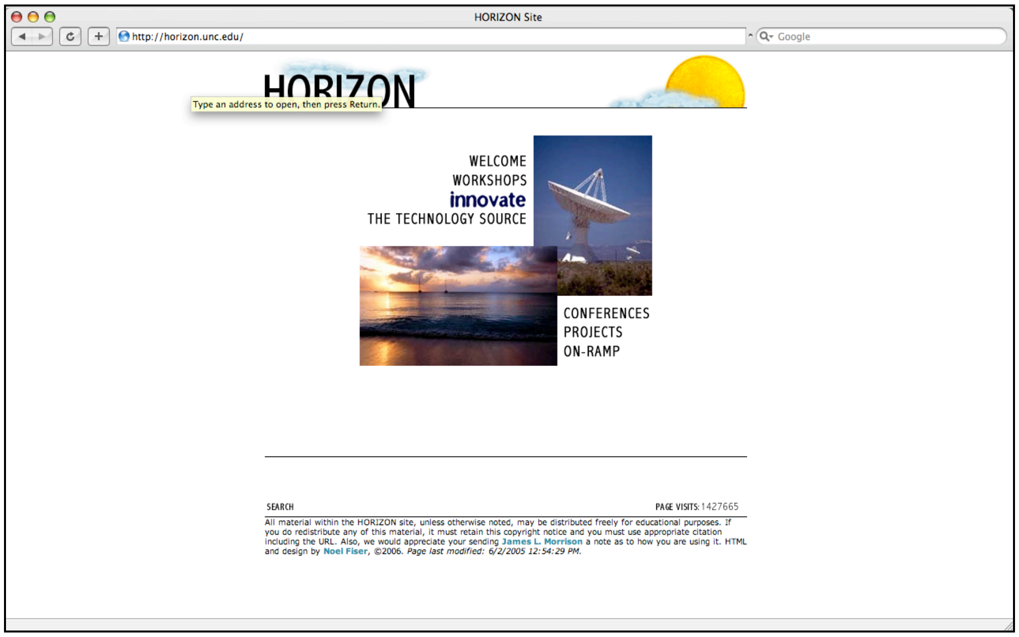
*Quisque sodales ante auctor ipsum. Vestibulum interdum euismod mauris.* Praesent sed urna. Donec ut orci. Nunc eu massa imperdiet felis lobortis venenatis. Praesent id arcu vel justo tempor aliquam. Nunc justo tortor, porta a, lacinia a, aliquet in, quam. Donec pede mi, hendrerit sit amet, ullamcorper eu, porta eu, turpis. Nullam scelerisque urna non massa. **Suspendisse at mi nec est** Sed viverra pede. Integer porta. In hac habitasse platea dictumst. Curabitur porta, leo dignissim mollis pulvinar, ante lorem condimentum erat, sed ornare lacus lorem ut orci. Donec et mauris.

Nullam et justo. Phasellus eleifend. Integer pharetra neque. Donec dignissim condimentum lorem. Donec tristique, odio in elementum fermentum, tortor massa eleifend nisl, in pulvinar magna pede non leo. Fusce ac nisi non orci gravida molestie. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Integer nec lectus. Maecenas blandit neque. Quisque fermentum, neque quis dignissim vulputate, velit lectus blandit ante, vel facilisis sem massa rutrum lectus. Nulla ut dui eu eros interdum commodo. Aenean accumsan feugiat nisi. Maecenas nisi. Praesent luctus. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Lorem ipsum dolor sit amet, consectetuer adipiscing elit.

**Mauris eleifend quam sit amet dolor.** Mauris velit risus, suscipit ac, molestie non, sodales eu, enim. Fusce enim. Aenean lacus neque, pulvinar vitae, hendrerit sed, convallis a, magna. Cras nec lectus. Donec aliquam sodales nisi. Cras vitae sem. In ornare, dui porttitor tristique aliquam, nisi velit interdum est, vel iaculis tellus neque non erat. Phasellus convallis, nulla id placerat tristique, nulla sapien egestas nisl, accumsan lobortis ipsum erat eu neque. Suspendisse potenti. Nullam dignissim risus in justo. Nullam mollis tortor at metus. Praesent nec tortor. Aenean sit amet nisi. Donec eget ipsum.

# Appendix B: Test sites

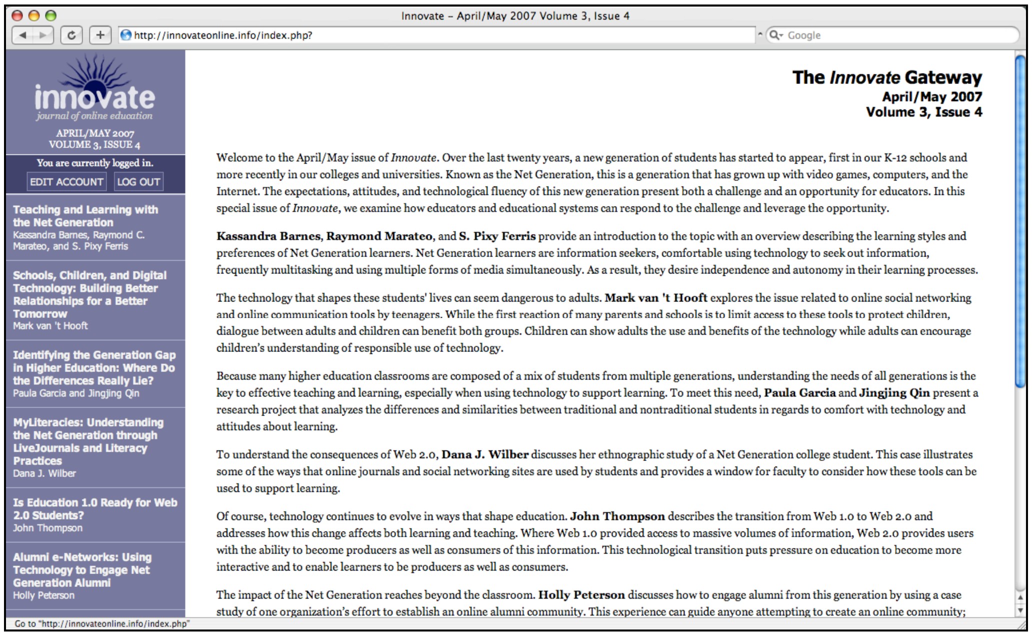
The Horizon Site (http://horizon.unc.edu/)



PAGE VISITS: 1427665

All material within the HORIZON site, unless otherwise noted, may be distributed freely for educational purposes. If you do redistribute any of this material, it must retain this copyright notice and you must use appropriate citation including the URL. Also, we would appreciate your sending **James L. Morrison** a note as to how you are using it. HTML and design by **Noel Fiser**, ©2006. *Page last modified: 6/2/2005 12:54:29 PM.*

*Innovate* (http://innovateonline.info/)


Innovate – April/May 2007 Volume 3, Issue 4
http://innovateonline.info/index.php?
Google
innovate
journal of online education
EDIT ACCOUNT
LOG OUT
Go to "http://innovateonline.info/index.php"

APRIL/MAY 2007
VOLUME 3, ISSUE 4

You are currently logged in.

**Teaching and Learning with the Net Generation**
Kassandra Barnes, Raymond C. Marateo, and S. Pixy Ferris

**Schools, Children, and Digital Technology: Building Better Relationships for a Better Tomorrow**
Mark van 't Hooft

**Identifying the Generation Gap in Higher Education: Where Do the Differences Really Lie?**
Paula Garcia and Jingjing Qin

**MyLiteracies: Understanding the Net Generation through LiveJournals and Literacy Practices**
Dana J. Wilber

**Is Education 1.0 Ready for Web 2.0 Students?**
John Thompson

**Alumni e-Networks: Using Technology to Engage Net Generation Alumni**
Holly Peterson

**The *Innovate* Gateway**
**April/May 2007**
**Volume 3, Issue 4**

Welcome to the April/May issue of *Innovate*. Over the last twenty years, a new generation of students has started to appear, first in our K-12 schools and more recently in our colleges and universities. Known as the Net Generation, this is a generation that has grown up with video games, computers, and the Internet. The expectations, attitudes, and technological fluency of this new generation present both a challenge and an opportunity for educators. In this special issue of *Innovate*, we examine how educators and educational systems can respond to the challenge and leverage the opportunity.

**Kassandra Barnes**, **Raymond Marateo**, and **S. Pixy Ferris** provide an introduction to the topic with an overview describing the learning styles and preferences of Net Generation learners. Net Generation learners are information seekers, comfortable using technology to seek out information, frequently multitasking and using multiple forms of media simultaneously. As a result, they desire independence and autonomy in their learning processes.

The technology that shapes these students' lives can seem dangerous to adults. **Mark van 't Hooft** explores the issue related to online social networking and online communication tools by teenagers. While the first reaction of many parents and schools is to limit access to these tools to protect children, dialogue between adults and children can benefit both groups. Children can show adults the use and benefits of the technology while adults can encourage children's understanding of responsible use of technology.

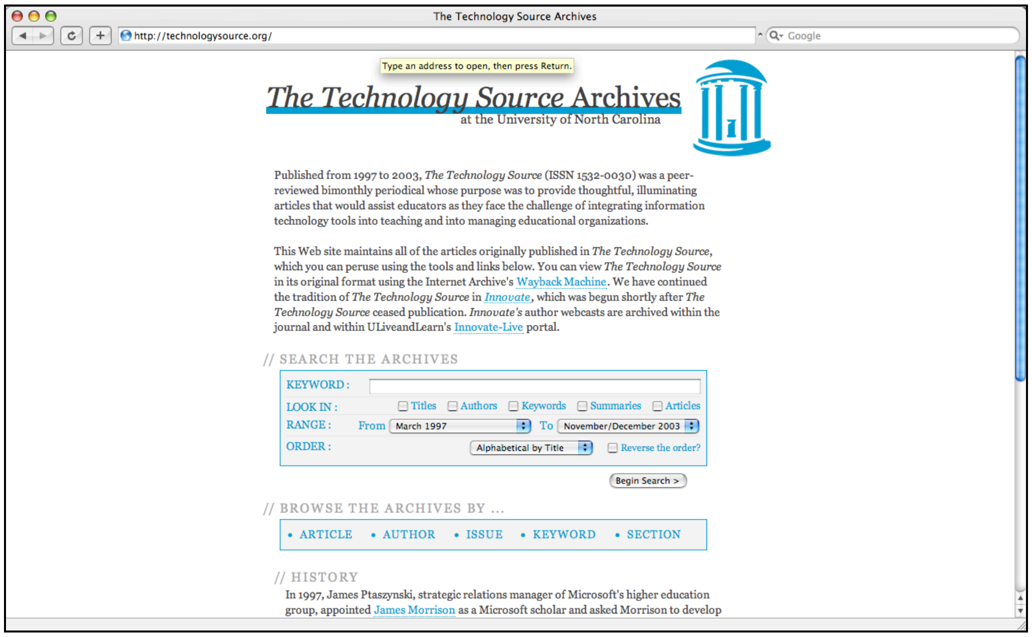
Because many higher education classrooms are composed of a mix of students from multiple generations, understanding the needs of all generations is the key to effective teaching and learning, especially when using technology to support learning. To meet this need, **Paula Garcia** and **Jingjing Qin** present a research project that analyzes the differences and similarities between traditional and nontraditional students in regards to comfort with technology and attitudes about learning.

To understand the consequences of Web 2.0, **Dana J. Wilber** discusses her ethnographic study of a Net Generation college student. This case illustrates some of the ways that online journals and social networking sites are used by students and provides a window for faculty to consider how these tools can be used to support learning.

Of course, technology continues to evolve in ways that shape education. **John Thompson** describes the transition from Web 1.0 to Web 2.0 and addresses how this change affects both learning and teaching. Where Web 1.0 provided access to massive volumes of information, Web 2.0 provides users with the ability to become producers as well as consumers of this information. This technological transition puts pressure on education to become more interactive and to enable learners to be producers as well as consumers.

The impact of the Net Generation reaches beyond the classroom. **Holly Peterson** discusses how to engage alumni from this generation by using a case study of one organization's effort to establish an online alumni community. This experience can guide anyone attempting to create an online community;

The *Technology Source* Archives (http://technologysource.org/)



The Technology Source Archives

http://technologysource.org/

Type an address to open, then press Return.

*The Technology Source* Archives

at the University of North Carolina

Published from 1997 to 2003, *The Technology Source* (ISSN 1532-0030) was a peer-reviewed bimonthly periodical whose purpose was to provide thoughtful, illuminating articles that would assist educators as they face the challenge of integrating information technology tools into teaching and into managing educational organizations.

This Web site maintains all of the articles originally published in *The Technology Source*, which you can peruse using the tools and links below. You can view *The Technology Source* in its original format using the Internet Archive's Wayback Machine. We have continued the tradition of *The Technology Source* in *Innovate*, which was begun shortly after *The Technology Source* ceased publication. *Innovate's* author webcasts are archived within the journal and within ULiveandLearn's Innovate-Live portal.

// SEARCH THE ARCHIVES

KEYWORD :

LOOK IN : ☐ Titles ☐ Authors ☐ Keywords ☐ Summaries ☐ Articles

RANGE : From March 1997 To November/December 2003

ORDER : Alphabetical by Title ☐ Reverse the order?

Begin Search >

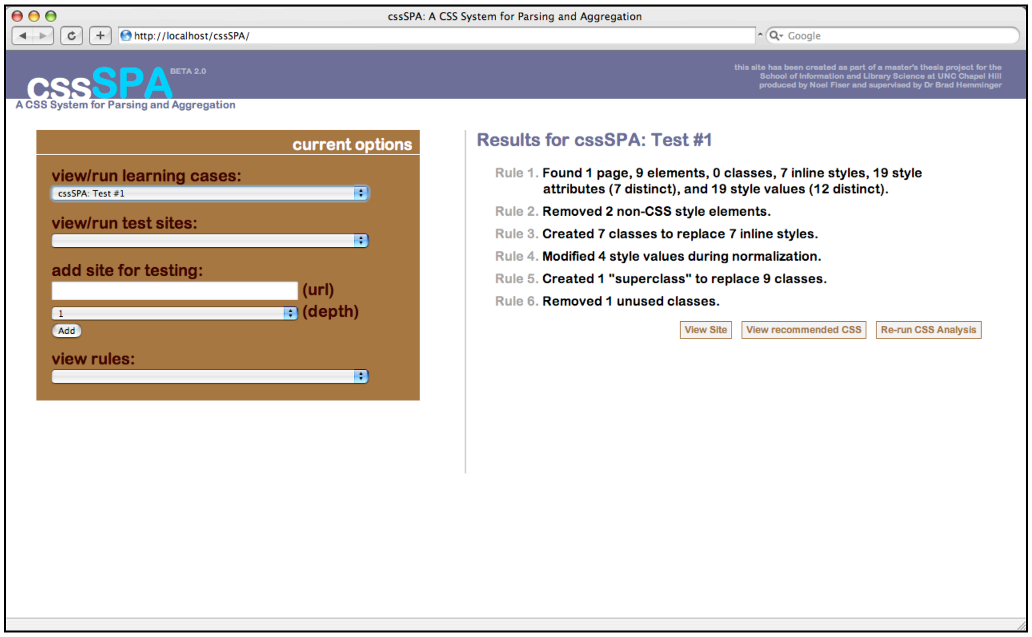
// BROWSE THE ARCHIVES BY ...

• ARTICLE • AUTHOR • ISSUE • KEYWORD • SECTION

// HISTORY

In 1997, James Ptaszynski, strategic relations manager of Microsoft's higher education group, appointed James Morrison as a Microsoft scholar and asked Morrison to develop

# Appendix C: The CSS Spa Web Site



cssSPA: A CSS System for Parsing and Aggregation

http://localhost/cssSPA/

cssSPA BETA 2.0

A CSS System for Parsing and Aggregation

this site has been created as part of a master's thesis project for the School of Information and Library Science at UNC Chapel Hill produced by Noel Fiser and supervised by Dr Brad Hemminger

current options

view/run learning cases: cssSPA: Test #1

view/run test sites:

add site for testing:

(url)

1 (depth)

Add

view rules:

Results for cssSPA: Test #1

Rule 1. Found 1 page, 9 elements, 0 classes, 7 inline styles, 19 style attributes (7 distinct), and 19 style values (12 distinct).

Rule 2. Removed 2 non-CSS style elements.

Rule 3. Created 7 classes to replace 7 inline styles.

Rule 4. Modified 4 style values during normalization.

Rule 5. Created 1 "superclass" to replace 9 classes.

Rule 6. Removed 1 unused classes.

View Site | View recommended CSS | Re-run CSS Analysis

The entirety of the CSS Spa Web site, including all PHP code and MySQL data definition language is available online at http://noelfiser.com/cssSPA/cssSPA.code.zip. A demonstration version of the user interface is available at http://noelfiser.com/cssSPA/demo/.

# REFERENCES

Andreessen, Marc. (1994, February 17). Indented <MENU>s. Message posted to the W3Cs WWW-TALK electronic mailing list, archived at http://ksi.cpsc.ucalgary.ca/archives/WWW-TALK/www-talk-1994q1.messages/643.html

Berners-Lee, T. (1999). *Weaving the Web: The original design and ultimate destiny of the World Wide Web.* San Francisco: Harper Collins.

Briggs, O., Champeon, S., Costello, E., & Patterson, M. (2002, May). *Cascading Style Sheets: Separating Content from Presentation*. Berkeley, CA: FriendsofED/Apress.

Dudek, D. T. & Wieczorek, H. A. (2003). A simple web content management tool as the solution to a web site redesign. *Proceedings of the 31st annual ACM SIGUCCS conference on user services.* San Antonio, TX: ACM Press, 179-181.

Raggett, D. (1996, May). *W3C Request for Comments: 1942, HTML Tables.* Retrieved January 25, 2007, from http://www.ietf.org/rfc/rfc1942.txt

Koch, P. (2007). *CSS contents and browser compatibility.* Retrieved February 3, 2007, from http://www.quirksmode.org/css/contents.html

Lie, H. W. & Bos, B. (2005). *Cascading Style Sheets, designing for the Web* (3rd ed.). Upper Saddle River, NJ : Addison Wesley.

Lie, H. W. & Saarela, J. (1999). "Multipurpose Web publishing using HTML, XML, and CSS." *Communications of the ACM* 42(10): 95-101.

Nardini, H. K. G., Linden J., Mayman, G., Reardon, K. & Shimp, A. (2002). "Lessons for working with Web designers."*Online 26*(2): 51-56.

Oakes, C. (1999, October 23). "Interview with the Web's creator." *Wired*. Retrieved January 25, 2007, from http://www.wired.com/science/discoveries/news/1999/10/31830

Reed, D., & Davies, J. (2006, February). “The convergence of computer programming and graphic design.” *Journal of Computing Sciences in Colleges* 21(3): 179-187.

Ricca, F., Tonella, P., & Baxter, I. D. (2001). Restructuring Web applications via transformation rules. *Proceedings of the First IEEE Workshop on Source Code Analysis and Manipulation,* 150-160.

Weiss, A. (2006). “The web designer's dilemma: when standards and practice diverge.” *netWorker* 10(1): 18-25.