HTML
in the context of
natural language variation and change

Sarah Tan
Senior Thesis
Linguistics Department
December 12, 2003

1.0 Introduction*

The World Wide Web is a major technological advance that has profoundly changed the way people live, in a manner that was unprecedented at the time of its invention in 1989. The underlying code that makes it work, that allows webpages to display properly is Hypertext Markup Language (HTML), which, naturally, is no older than the environment for which it was created. Over the course of almost fifteen years—a substantial amount of time in the context of hi-tech advances—HTML has faced issues of variation and standardization that fit in with natural language change and evolution. I will make the argument that HTML allows us insight into the process of language variation and is a case in point of language change inevitability.

HTML has expanded from a simple markup language to include new features and visual formatting capabilities rather quickly by any measure, much less from the viewpoint of a historical linguistics field that covers decades and centuries at a time. It can be seen as language evolution in a highly compressed nutshell. I will outline the reasons for and implications of this phenomenon.

Section 2 covers the origins of HTML and the mentality under which the World Wide Web was conceived by Tim Berners-Lee that deliberately made HTML more flexible and open than computer programming languages and more like natural language. Section 3 details some common language change phenomena that HTML has faced or

* Thanks are in order for everyone who has been involved in this process, including my adviser Ted Fernald, my student readers Rachel Fichtenbaum and Joshua Anderson, and all the other seniors in my section of the research seminar, especially those who indulged my penchant for misery poker. Douglas Blank, my second reader, contributed invaluable knowledge and advice down to the wire. Eric Raimy made himself available as a sounding board for my ideas and concerns. Jessie Johnston was my default on-campus linguistics support. Dave McOwen introduced me to the code and back-end of the Web. Numerous friends provided much-appreciated encouragement, reassurance, patience, stress relievers, late night conversations, and good vibes. Among them, Tina, Meredith, Shweta, Carol, Karen, and the Heathers deserve special mention. My parents are in the background of everything I do. All errors, of which there must be plenty, remain mine alone.

undergone, such as lexical and semantic change and language planning, as well as mention of similar phenomena which HTML cannot undergo because of its inherent differences with natural language. This section includes the problems unique to HTML from being reliant on browsers to render it. Section 4 explains the impact of the Web standards movement and the direction in which the World Wide Web Consortium (W3C) is steering HTML, and the increasing compliance of the major browser companies to Web standards. Section 5 examines the future of HTML as the stricter XHTML and possibly the end of the linguistic free-for-all that characterized the state of HTML in the 1990s. Section 6 draws the conclusions and analyzes this moment in history when a computer language acted just a little bit human.

2.0 Markup language precedents—SGML

For our purposes, the story begins with Standard Generalized Markup Language (SGML), a powerful markup language that had been invented in the 1960s to combat "the problems of transporting documents across different computer systems (Aronson 1994: 2). By the time that Tim Berners-Lee was coming up with the idea of the World Wide Web and hypertext documents some twenty years later, SGML was "already preferred by some of the world's documentation community and at the time considered the only potential document standard among the hypertext community" (Berners-Lee 1999: 41). SGML was in fact officially adopted as a standard by the International Standards Organization (ISO) in 1986.

However, Berners-Lee wanted his new information system to be used by people globally, and SGML was too complex for his purposes. Knowing that he would need the support of existing community in promoting the Web, he therefore made the political

decision to develop HTML so that it looked enough like SGML. That way, people who already knew SGML would be able to use HTML with minimal transitional problems.

2.1 Format

The main feature that Berners-Lee kept from SGML was "a simple system for denoting instructions, or 'tags,' which was to put a word between angle brackets" (Berners-Lee 1999: 41). The word or characters between the angle brackets is called an element and usually consists of a start tag and an end tag, such as `<h1></h1>`, which marks the text contained within as a level-one heading. Marking up text by putting it between the tags, as in `<h1>Warning</h1>`, will classify the word "Warning" as the main heading of the document. Although "tag" is frequently used metonymically to mean "element," I will try to keep the two terms distinct to avoid ambiguity.

Berners-Lee also tried to maintain consistency by identifying the particular set of elements employed by SGML and having HTML use the same ones wherever possible. He said that he "did clean up the language a certain amount, but it was still recognizable" (42). People having a sense of comfort and familiarity about the language before they even started was just as important as the technical aspect, where HTML was easier to use than SGML.

2.2 Structure

By convention, markup languages focus on the structural aspects of a document and are not directly involved in the visual representation of that structure ("Markup language"). Tags are meant to represent the function that the encompassed text serves in the context of a particular document, which in itself has no effect on how text looks. Berners-Lee originally only wanted to construct a network for communication and

information sharing between physicists and "viewed his creation as a medium for the easy exchange of text documents; therefore, he included no typographical control in the structured language of HTML" (Zeldman 2003: 300).

However, lack of typographical control does not mean that all text in an HTML document looks the same. Certain HTML elements have visual properties associated with them; for example, headings, which are `<h1>`, `<h2>`, `<h3>`, `<h4>`, `<h5>`, and `<h6>`, all automatically start on a new line and are bolded. They also have size properties, starting at the biggest with `<h1>` and decreasing in size until `<h6>`. For HTML, the separation of style and content was an important philosophical rule, as "this was the only way to get it to display reasonably on any variety of different screens and sizes of paper" (Berners-Lee 1999: 41).

Because HTML pages are accessed by individuals on their own devices, there are physical variables, such as the size of a screen; setting variables, such as screen resolution; and interpretation variables, such as the choice of software. There are no comparable issues with other, older forms of media like books, television, or radio, all of which have a prepackaged output and a fixed method of delivery.

2.3 Attitude

The key to the World Wide Web functioning the way that Berners-Lee envisioned was a complete decentralization. This was necessary not only for technical viability, since the system would have a limit on scale growth if there were some kind of main node (Berners-Lee 1999: 16), but also because of the type of open mindset people needed to have to sustain it. The Internet and World Wide Web were the first signs of the open-

source[1] culture that has become much more common in today's software development circles. Berners-Lee's view was that "making collaboration work is a challenge. It is also fun, because it involves the most grassroots and collegial side of the Web community." As such, all of his code, since 1991, has been open-source software (171).

Additionally, once the need for clients[2] for different platforms became apparent for the use of the Web to really spread, he "energetically suggested to everyone everywhere that the creation of browsers would make useful projects for software students at universities" (56). While this was undoubtedly in part because Berners-Lee did not have the time or resources to develop all these browsers himself, a conventional software corporation would probably have been willing to delay the expansion of the Web for as long as it took for its own programmers to develop the products.

In 1993, there was an incident of industry-wide impact where the University of Minnesota announced that it would ask for a license fee from certain users[3] of its gopher information system, which was similar to the Web. Even though this would only apply to server software, not to browsers or the act of browsing, and only to companies, not nonprofit or educational institutions, using gopher was now too risky, and the academic and Internet communities abandoned it (Berners-Lee 1999: 72-73). So as not to meet a similar fate, Berners-Lee convinced his employer, CERN (now called the European

[1] Open source is the idea of a person or company releasing the source code for a piece of software to the public so that other computer programmers can make suggestions, fix bugs, or even modify and redistribute the code themselves. It has the general advantage of producing better products with faster improvements. Most open source software is free, although this is not necessarily the case ("Open Source" 2003).
[2] A client is piece of software that will access another computer on a network. For the Web, the client is called a (Web) browser—it accesses the server on which the requested HTML or other file is stored and displays the file on screen in a form that can be read by people. Henceforth, I will use the term "browser" for this particular type of software.
[3] "Users" or "end users" in this context are any people who navigate the Web, using a browser. The terms tend to imply that they lack technical knowledge of how the Web works. Users are distinguished from "(Web) developers," people who professionally create webpages and presumably keep abreast of new developments.

Laboratory for Particle Physics) in Geneva, to have Web technology put in the general public domain, with no strings attached. This was even more lax than the GNU Public License that he had been pushing for before, and now, anyone could "use the Web protocol and code fee of charge, to create a server or a browser, to give it away or sell it, without any royalty or other constraint" (Berners-Lee 1999: 74).

2.4 Implications

Berners-Lee's decision to open the field for browser development had the result of also opening up the definitions for the technologies dependent on being accurately understood by browsers. Aside from HTML, these technologies included Hypertext Transfer Protocol (HTTP) and Uniform Resource Identifiers (URIs),[4] which brought up the issue of uniformity. "As the Web slowly spread around the world, [Berners-Lee] started to be concerned that people who were putting up servers would not use HTTP, HTML, and URIs in a consistent way. If they didn't, they might unintentionally introduce roadblocks" to the process (53).

This was a legitimate worry, though at that point in the early 1990s, irregularities had not yet appeared. The specifications for URI and HTTP were each successfully standardized. These were also the two that were most important to have officially standardized (and have the specifications followed in practice) in order for the Web to work (Berners-Lee 1999: 36). HTML, however, would soon be subject to various forces that would extend and expand it to a point of barely being recognizable from its original form.

[4] HTTP defines the rules that computers use talk to each other, and a URI is an address format for the type of application protocol being used and for the location of the computer with the desired data (Abbate 1999: 215, Berners-Lee 1999: 36). Neither is relevant enough to the discussion to warrant more details than what has been stated here.

These modifications to HTML were largely a result of the way that browsers developed and the massive number of users, although there are certain phenomena that transpired because of the way that people created and interacted with HTML code. For the rest of the 90s, HTML had so few inviolable restrictions and regulation that it behaved flexibly, developing and changing more like natural language than a computer language. The next decade would consist of multitudinous variations that many people now view as characteristic of that time.

3.0 Linguistics and Markup Language

This section covers the changes in HTML that I will argue make it fit into the model of natural language variation. For each analysis of a linguistics phenomenon that has affected HTML, I will first provide a brief overview of the processes as they occur in natural language. As HTML is obviously a computer language and not a natural language, I will then take some time to explain the technological factors that have influenced its development. In some cases, the technological aspects have also shielded HTML from being affected by more types of language change. After that, I will examine the changes that have affected HTML and provide examples that demonstrate its similarity to natural language in the way that HTML can differ from its earlier states.

3.1 Lexical change

There are a number of ways that new words appear in natural languages, some of which are common, and others of which are less so. One of the most widespread practices is borrowing of words from other languages. There are two main reasons for this—one is because the existing language has no equivalent word for the concept, such as “ski,” a loan word into English from Norwegian. The other is prestige, which is the reason for the

massive influx of French loan words into English at the time of the Norman invasion of England in the eleventh century that resulted in English words like "courage," "music," "pity," and "language," just to name a few (Trask 1996: 18-20).

Even without the influence of borrowing from other languages, formation of new words also happens by **compounding**, where two existing words combine to create a new one ("lipstick," "skinhead"); **derivation**, where new words are created by adding affixes ("establish" can become "antidisestablishmentarianism"); and **initialism**[5], where a phrase is shortened to the first letters of the important words, among others (Trask 1996: 36). The number of initialisms and acronyms is increasing with computer technology names like CPU (Central Processing Unit), DOS (Disk Operating System), and, of course, our beloved HTML. Finally, "one of the rarest of all ways of obtaining new words is simply to invent them, more or less out of thin air." One example of lexical invention that Trask cites is the English word "blurb," which was created to describe the paragraph on the back of a book containing praise and a synopsis (Trask 1996: 37).

3.1.1 Brains and Browsers

The largest difference between natural language and computer language that affects their development process is the method of interpretation. Since natural languages are spoken or otherwise communicated directly from one person to another, the only cognitive devices involved in the exchange are the two people's brains. For computer languages, however, there is a multi-step process because a piece of software has to read

[5] Specifically, "initialism" is for terms that are pronounced by the letter, like IPA (International Phonetic Alphabet). "Acronym" is for terms that are pronounceable as words, like NATO (North Atlantic Treaty Organization). Speakers sometimes forget what acronyms stand for and that they were not originally words, such as "scuba" (self-contained underwater breathing apparatus), which is also no longer written in capital letters (Trask 1996: 36).

the code and produce human-readable results from the input. In fact, since humans write the code in the first place, they have to produce machine-readable code, which the software then has to convert back to human-readable content. Thus, there are two main places where the translation process can go awry—first in the code creation process, and second in the machine rendering of the code.

This scenario is accurate for computer languages in general, though of course here we are limiting the type of code to that written in HTML, which has its unique set of problems, different from natural language and also from other computer languages. The crux of the distinction is the browser, which behaves differently from traditionally and notoriously strict compiler software that translates computer programming languages[6].

3.1.2 Browser Development

As mentioned before, Berners-Lee made the strategic decision to allow anyone to create a browser for viewing webpages. Since the rules built into the browsers determine the allowable HTML, this had the effect of allowing programmers to come up with new and different elements that would be accommodated in their individual browsers. In effect, this gave them the power to add items to the lexicon of HTML that anyone with the appropriate browser would be able to understand correctly.

Indeed, several programmers offered ideas for augmenting HTML's capabilities with new elements by coding them in their own browsers. Some of the ideas proposed then, such as being able to include images or display information in tables, would eventually be officially adopted. Being able to have a browser implement items that were not part of existing HTML necessarily meant extending HTML, and so programmers

[6] Programming languages are used to write instructions in a way that a computer can understand, whereas markup languages are used to communicate the structure and formatting of documents to a computer.

simply made up names of elements according to what they thought made sense. This was not without precedent, as in SGML there was the built-in ability to define a new element in a separate file. Since, as part of the simplification process, HTML did not directly support this behavior, programmers had to define their new elements into the browser engine itself, the equivalent of reconfiguring the machine brain instead of just being able to feed it a new rule or two.

To Berners-Lee, it seemed clear that "there was growing competition among the browsers, even if it was on a small scale. Many of the people developing browsers were students, and they were driven to add features to their version before someone else added similar features. They held open discussion about these things […], preserving the open social processes that had characterized Internet software development. But there was still an honorable one-upmanship, too" (Berners-Lee 1999: 67). At this point, circa 1993, however, these additions to HTML tended to be responses to what the small Web community expressed a desire to see in a browser.

3.1.3 Netscape Reigns

The way in which new elements were added to HTML changed with the popularization of the Web and the coinciding commercialization of the browser industry. The first browser to emerge as the leader in its field was called Mosaic, which came out in Nov. 1993. It supported images, was easy to download and install, sported a user-friendly, point-and-click interface, and was available for multiple operating systems, all of which boosted its popularity[7] and raised the bar for future browsers (Berners-Lee

[7] In fact, the Mosaic browser picked up so quickly that some people started using "Mosaic" to refer to the Web (Berners-Lee 1999: 70) as a result of a public relations push from the National Center for Supercomputing Applications (NCSA), which developed Mosaic. This is a nice side example of brand names becoming generic terms and how naturally such linguistic phenomena happen (Trask 1996: 36).

1999: 68, Koch 2003). After some internal organizational changes, Mosaic was renamed Netscape, and the next year Netscape released its new browser, Navigator 1.0, which was significant because Netscape released it over the Internet and didn't charge for it. "Within several months, the majority of people on the Web were using it" (Berners-Lee 1999: 99).

New releases of Netscape continued to provide support for new features, meaning that with every release of a different version, it created new HTML elements that produced certain effects. From 1994 to 1997, Netscape became the standard Web browser, and at its peak around 1996, Netscape became the most popular computer program with 38 million users, which was upwards of seventy percent of the market share (Berghel 1999, Castro 2000: 14). It accomplished this by maintaining the strong user base and brand name recognition and also by keeping the browser free for non-commercial use (Koch 2003).

3.1.4 Enter Microsoft

Although Microsoft had not been involved in the Internet or the Web before, it saw this new and emerging field in which it could make an impact and decided to enter the browser business in 1995. After two false starts, Microsoft produced its first decent browser in Internet Explorer 3, which started to gain market share, and by 1997 was used by about 35 percent of users, decreasing Netscape's users to just under 60 percent of the total (Berghel 1999).

The next two years were characterized by what became known as the "Browser Wars" between Netscape and Microsoft. Both companies released their respective version 4 browsers in mid to late 1997. At this point, both browsers were available to all

users at no cost[8], and the goal was to obtain (or in Netscape's case, regain) a monopoly. To this end, all kinds of corporate tactics were used to gain an edge over the competitor in a less than honorable one-upmanship.

3.1.5 Going Head to Head

In the struggle for users, programmers built in all kinds of features into the browsers. The logic went that if there was a new feature that only one browser supported that more users wanted to make use of, that browser and company would win (Castro 2000: 14). Although Netscape had been creating proprietary extensions for some time that would not work in other browsers, the impact was relatively small because most people used Netscape. With Microsoft now a serious contender and creating its own set of extensions, two definite camps for each side emerged, and what ensued was a corporate battle based on what HTML would work in which browser that would change the way people had to write HTML.

3.1.6 HTML Lexical Source

In HTML, the only lexical entries are the elements, the attributes of elements, which define options pertaining to that element, and the values of attributes[9], which specify desired choice. By virtue of the fact that Berners-Lee and the inventers of SGML[10] were all English-speakers, both SGML and HTML are based on English. In deciding names for most HTML elements, then, Berners-Lee simply picked appropriate English words that accurately described an elements function.

---

[8] Netscape had originally made its browser free for educational purposes, but charged a small fee for corporate users. This changed in 1997 under the pressure of Explorer being free of charge to everyone (Koch 2003).

[9] For example, the tag, `<p align="center">` will make the text following it into a centered paragraph. `p` is the element, and `align` is the attribute of `p`, and `center` is the value of `align`. Attributes are always optional.

[10] Charles Goldfarb, Edward Mosher and Raymond Lorie, whose initials formed the original initialism for SGML, which was the reason that Goldfarb coined the term "markup language" (Anderson 2001).

For programming purposes, it doesn't matter what the actual names of items are; in theory, names could be any random mix of letters, numbers, and certain other special characters. Common sense, however, leans toward sensible, semantically relevant names. As such, HTML has the structure of an SGML document, but it borrows all of its lexical items from English. This is as much happy coincidence of the nationalities of the inventors of these two markup languages as it is a sign of English's role as an increasingly prevalent global language.

3.1.7 HTML Initialism

For purposes of practicality, programmers needed to keep names relatively short because each additional character took up an additional byte of space, which could easily lead to unnecessarily large file sizes. On top of simple abbreviations and clipping for the majority of element names (e.g. `<img>` for images, `<div>` for divisions), another method of condensing meaning into a few letters was by initialism.

The full names of some HTML initialisms are better remembered than others. For example, `<ol>` is an ordered list and `<ul>` is an unordered list, which is relatively intuitive because of the elements' usage and visual properties—by convention, ordered list items display prefaced by ascending numbers, and unordered list items display prefaced by a bullet. On the other hand, `<td>` marks a table cell and is memorized as such; it is never described as the "tabular data" tag, even though it is still pronounced as the [ti.di] tag. Therefore, just as people know what a laser is even if they forget (or never know) that it originally stood for "light amplification by the stimulated emission of radiation" (Trask 1996: 36), the same thing is happening with certain initialized HTML element names.

3.1.8 HTML Reinvented

Although invention is the least common way to obtain new words in natural language, we have already seen that it is the main way by which HTML has been expanded and changed. Essentially, the original HTML elements marked basic structure, rudimentary graphics, and hypertext links (Berghel 1999). The tables below show a sampling and timeline of new elements that Netscape and Microsoft introduced to HTML during the Browser Wars (Castro 2000:101, 362-367; Zeldman 2003: 301):

Table 3.1: Netscape Extensions

Name	Description
`<blink>`	Flashing text
`<layer>`	Positioning Element
`<font size…>`	Size of text

Table 3.2: Microsoft Extensions

Name	Description
`<marquee>`	Moving text
`<iframe>`	Floating frame
`<font face…>`	Font of text

Table 3.3: Different ways of doing the same thing

Browser	Name	Description
Netscape	`<body marginwidth="x" marginheight="y">`	Side and top and bottom margins of page
Microsoft	`<body leftmargin="x" topmargin="y">`	

3.1.9 Lexical changes' inapplicability to HTML

We have seen that it is not difficult to create extensions to HTML; however, modifying existing elements is practically unheard of because it would go against the guideline of backward compatibility and cause documents written according to the previous specification to be incompatible with new browsers. This state of things limits the possible lexical changes to HTML; consequently, many of the changes that affect natural languages in somewhat mysterious and inexplicable ways simply do not apply. HTML could not and did not carry over English's morphological system with the lexical borrowings, so it has no affixes to make derivation possible. Elements cannot simply combine or attach to one another to form compounds. What is notable, however, is that

even though not all language change phenomena apply to HTML, there are actual indications of language change in HTML.

3.1.10 Structure vs. Presentation

While Microsoft and Netscape were busy trying to beat each other out by adding features to their browsers, they completely disregarded any concern for separation of style and content. For example, `<blink>` and `<marquee>` were competing elements, and aside from that fact that they were of dubious benefit for the health and sanity of the Web, neither provide any information about the role that blinking or marquee-ing text played in the structure of a document.

By this point, however, the market had changed, and the audience was no longer limited to a technologically adept and exacting community. “In terms of acceptance, the Web got off to a faster start than any other medium ever introduced” (Zeldman 2003: 22), and most users neither knew nor cared that HTML was no longer sticking to its technical definition.

3.1.11 Proprietary Problems

Extending HTML in itself was not an issue; however, because each browser refused to support the extensions that the other introduced, this is how the issue of lack of mutual intelligibility came to be. The late 1997 4.0 browsers from Netscape and Microsoft were completely incompatible with each other. “They were also completely incompatible with previous versions of themselves (what worked in Netscape 4 would not work in Netscape 3), not to mention being utterly incompatible with ‘off-brand’ browsers that meekly supported basic specs like HTML instead of making up their own languages and attributes” (Zeldman 2003: 38). It was equivalent to a new generation

defining its own group membership by the use of slang. HTML, then, was faced with a messy and divisive future of two somewhat antagonistic strains that were each matched with only one browser. It was this state of affairs that would soon lead to the movement for fixing HTML's plight.

3.2 Semantic Change

Change in meaning is another common type of language change. "English words have been changing their meanings for centuries, and words are still changing their meanings today" (Trask 1996: 37). Semantic shifts often garner the most irritation from grammarians of the prescriptive school, who take it upon themselves to campaign for preserving the "right" meaning. One case that is in the process of completing a shift is the use of "hopefully" as a sentence adverb (8), which draws the ire of more conservative English speakers while being seen as nothing out of the ordinary to younger speakers who have grown up with this structure.

Misunderstandings are another way by which people place different meanings on words than what they originally meant. This can happen when a word is "commonly used in a context in which a different interpretation of the whole sentence is possible and reasonable" (Trask 1996: 38). "Momentarily" is increasingly used to mean "in a short while" or "soon," though the Oxford English Dictionary lists this as occurring chiefly in American English. The older meaning of "momentarily" is "for a short while," but in a sentence like "She will go momentarily," both meanings could be appropriate, which is likely how they became confused.

Yet another normal way for words to change meaning is a change in the world (Trask 1996: 38). This type is particularly relevant with the emergence of new

technologies that change the world. When automobiles were invented, the term used to refer to them was "motor car," since a "car" was a cart or wagon drawn by animals rather than being self-propelled. Before long, however, "motor cars" became much more popular and commonplace than old-fashioned "cars," and an English speaker today probably does not know the previous incarnation of the word "car" (38-39).

3.2.1 Adapting HTML Elements

Up to this point, we have looked mainly at how computer programmers have changed HTML by modifying the browser. While programmers change what is possible with HTML, HTML authors change how the language is used in much the same way that natural language speakers change their language over the course of time. For example, the `<table>` element was one of the earliest additions to the HTML specification to be able to display data in a tabular format. As the Web became more and more of a visual medium and users' expectations of webpage design increased, Web designers found that they could use tables for layout purposes, which allowed them more control over placement of elements on the webpage than they were previously afforded.

This practice became so widespread that "layout tables" became—and remain—the primary design method for a majority of today's Web designers. Since tables were established elements in HTML, they are rendered consistently in different browsers (Zeldman 2003: 193), and Web designers used this to their advantage to be able to reliably deliver more visually creative webpages. This practice became so common that many new Web designers did not realize the original purpose of tables, but rather thought that the main purpose of tables was in fact for layout. In this sense, the `<table>` element

underwent semantic change because of a change in the HTML design world, where layout tables become more common than data tables.

3.3 Interpretive Flexibility

3.3.1 Human Brain Adaptability

A speaker of any language can make a great deal of leeway in grammatical, syntactic, lexical, or speech production slips because the listener will still be able to comprehend what is being said, and the overall communication will still be a success. In other words, one does not have to be perfectly fluent in a language to effectively convey an idea. The human brain is capable, and one might even say adept, at making sense of spoken and written language when it deviates from the norm. This skill becomes apparent when people can make sense of intended meaning despite physical obstructions, like chewing gum or food in the speakers mouth, and also with children's speech, when they are learning but have not yet mastered the grammar of a language.

This phenomenon has been the subject of study by cognitive scientists and phonologists and is not within the scope of this paper; suffice to say, that it is far easier to program software that only has to behave within a strict set of rule. It is only recently that programmers have been able to successfully create software that can independently identify patterns and adjust behavior accordingly. Programmers are still in the early stages of improving on and refining the technology. Naturally, the discrepancy between those aspects of natural and computer-readable languages is less because of the nature of the languages themselves and more to do with the interpreter in question for each type of language.

3.3.2 Laxity in Browser Rendering

Neither Explorer nor Netscape demanded the strict adherence to syntax that programming language compilers do. This may have been intentional back when Berners-Lee wrote the first Web browser WorldWideWeb (later renamed Nexus), and it did provide certain advantages, like the ability of a non-techie to put together a webpage with only basic guidance on HTML. "This looseness makes it easy for anyone to create a Web page, even if they don't quite know what they're doing—and that, of course, was the idea." (Zeldman 2003: 102). This turned out to be a double-edged sword: on one hand, the simplicity of HTML that enabled the fast growth of the Web (Fensel 2003: 8), empowering millions of people. On the other hand, "the lack of uniform rules in HTML impedes data repurposing" now that pages are viewed on devices that weren't even around at the start of the Web (Zeldman 2003: 103).

Additionally, Explorer and Netscape treated ill-formed code differently—even though imprecise code (sometime called "tag soup") would still usually run, there might be some unexpected and different glitches in the webpages when they were viewed in either browser. One common example involved missing structural closing tags. If, for example, a table had the opening `<table>` tag but the closing `</table>` tag is accidentally omitted, Explorer would display the page perfectly, while Netscape would omit both the table and everything after it in the HTML document (Castro 2000: 325). These kinds of significant incongruities were regarded as typical of the frustrations that developers faced on a daily basis, who then had no alternative but to tackle browser inconsistencies as they were discovered individually.

Zeldman complains about how mainstream browsers of the time were "lax to the point of absurdity" and draws inevitable comparisons with programming languages like

C and Java, which "don't merely encourage proper coding practice [but] demand it […] In HTML, you must close some tags, mustn't close others, and might or might not want to close still others, depending on your mood" (Zeldman 2003: 39-40).

3.3.3 Coding Practices

Not all of the blame can be placed on browsers. Lack of competence on the part of HTML authors, even those who were professional developers, was more difficult of a problem to resolve. Those who were used to being able to write tag soup and have it display correctly only exacerbated the problem (Zeldman 2003: 40).

Berners-Lee actually had not anticipated this to be an issue when he defined HTML because he had thought that most people would not have to see the markup or know how to recreate it. As it turned out, the fact that HTML contained recognizable words differentiated it from lower-level assembly languages and peaked users' interests:

> "I never intended HTML source code (the stuff with the angle brackets) 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. But the human readability of HTML was an unexpected boon. To my surprise, people quickly became familiar with the tags and started writing their own HTML documents directly" (Berners-Lee 1999: 42).

3.3.4 What You Think You See is What You Think You Get

Part of the reason that so many webpages contain sloppy HTML is that the authors buy software with a user interface that lets them design webpages without having to hand code a text file. These WYSIWYG (What You See Is What You Get) editors usually produce pages that work fine, but the code they produce is not always the cleanest or most sensible. The latest versions of leading WYSIWYG editors like Macromedia Dreamweaver and Adobe GoLive have greatly improved their HTML output, which is

less important for the Web developers who already know HTML than it is for the "semi-skilled workers who would be powerless to create even a basic Web page if denied access to said tools" (Zeldman 2003: 78).

Then there are the people who don't know HTML, don't have access to or know how to use a WYSIWYG editor, and want to put up a webpage. They see that in Microsoft Word there is an option, "Save as Web Page." They click that, and everything works. It isn't until we look at the code that we see the difference. (1) below is actually a selection from my first website, for a class project in April 2002.

(1)

```
<p class=MsoNormal><a href="./glos.html">Venetian-Italian
Dictionary HTML</a></p>

<p class=MsoNormal><![if
!supportEmptyParas]> <![endif]><o:p></o:p></p>

<p class=MsoNormal><a href="./glosxml.txt">Venetian-Italian
Dictionary XML</a></p>

<p class=MsoNormal><![if
!supportEmptyParas]> <![endif]><o:p></o:p></p>

<p class=MsoNormal><a href="./images/index.html">Scanned
Images</a></p>

<p class=MsoNormal><![if
!supportEmptyParas]> <![endif]><o:p></o:p></p>

<p class=MsoNormal>Nazari, Giulio. <strong><u><span style='font-
weight:normal'>Dizionario veneziano-italiano e regole di
grammatica, ad uso delle scuole elementari di
Venezia</span></u></strong><strong><span style='font-
weight:normal'>.</span> </strong>Belluno: Tipografia Tissi,
1876.</p>
```

By using word processing software made by Microsoft to create a webpage, this excerpt became a case in point of proprietary, Microsoft-oriented code. It's not pretty, there is a disproportionate amount of extraneous markup, and there is no reason for meaningless drivel like `<![if !supportEmptyParas]>`, which doesn't even have an

element in the tag. In fact, the HTML that Word produces is so notoriously messy that Dreamweaver has an automated function, “Clean up Word HTML” for people who are editing or converting their old webpages.

This method has the additional impact of forcing other browser developers make their browsers either recognize the obfuscated markup or risk losing users to a Microsoft browser that will display the page the way the author wants. The person who uses a word processor to create HTML documents does not care at all about the inner workings of how exactly their site looks the way it does. From the corporate point of view, being able to invisibly control the code output increases the company’s leverage and influence in the hope for the proprietary method to become the new standard by default.

I redesigned the site in July 2003, some months after I learned HTML. The same excerpt is shown in (2):

(2)

```
<p><a href="glos.html">Venetian-Italian Dictionary HTML</a></p>

<p><a href="glosxml.txt">Venetian-Italian Dictionary XML</a></p>

<p><a href="images/index.html">Links to scanned images</a></p>

<p>Nazari, Giulio. <em>Dizionario veneziano-italiano e regole di
grammatica, ad uso delle scuole elementari di Venezia</em>.
Belluno: Tipografia Tissi, 1876.</p>
```

This time, the HTML is much more readable even to the human eye. There is less total code, so it will load that much faster. One of the biggest challenges for the Web development community is convincing software makers to program their new versions of WYSIWYG editors to create well-formed HTML. That way, people do not have to know anything about how HTML works but can still have their software do the work correctly for them.

4.0 Rules, Rules, Rules

The various language changes covered so far have involved naturally occurring processes that happen just because language is used. Not all changes are like this, however, and "it is quite possible for speakers to make deliberate decisions about the future of their language, and sometimes even to succeed in imposing decisions made by official bodies upon the speech and writing at least of educated speakers" (Trask 1996: 330). Trask qualifies his statement by limiting the influence of language planning organizations to educated speakers because they are more likely to pay attention and find merit in consciously adjusting their language according to formal rules.

The situation is quite the opposite for programming languages, where the norm is that an organization or even a single person is ultimately be in charge of the next direction of a language. HTML was not under the direct control of any such body, but that does not mean that one did not exist. In fact, we will see in this section the growing influence that language planning and standards is having on HTML.

4.1 Language Associations

Attempts by formal organizations to stipulate the direction of a natural language surface from time to time. Usually the goal is to stop the decline of a language and preserve its current (or even previous) state for the good of some well-meaning but often misguided goal, whether it be the language, the country, the culture, or another worthy ideal.

Aside from his day job as a satirist in the 1700s, Jonathan Swift was also an advocate of repairing what he viewed as the ongoing corruption of the English language. He proposed "an 'English Academy' [that] should be set up to fix English once and for

all, like a dead butterfly in a specimen box, after which nobody would be allowed to introduce any further changes at all, apart from the acceptance of an occasional new word which might be deemed necessary and allowable by the authorities" (Trask 1996: 6). As we know, of course, no English Academy was ever established, and even if it had been, it would not have had anywhere close to the kind of effect that Swift wished for. We can be quite sure of this because of a living example in the modern French language.

In 1570, the French created the Académie Française, "an august official body charged with making regulations for the proper use of French and staffed by distinguished (and often elderly) scholars of impeccable reputation. At frequent intervals the members of the French Academy meet to discuss things and to hand down solemn rulings about what French-speakers are allowed to say" (Trask 1996: 10). The sway that the Académie has on everyday speech can be seen in an example from June 2003. The Académie officially banned the word "email" in favor of the more French "le courriel," a combination of "courrier electronique" (electronic mail), in the latest of ongoing attempts to limit the "incursion of English words into the French lexicon" (Associated Press). Given that the Académie has not had success in eliminating assimilated English loan words like "le shopping" and "le weekend" from French conversation, it is probably safe to predict that "le courriel" will not dislodge "email."

This is not to say that language planning never works; however, the circumstances under which it is successful are generally limited to liberation from foreign domination, such as Finland's independence from Russia at the beginning of the twentieth century, or a pragmatic need for communication, such as with the unification of China in the second

century B.C. (Trask 1996: 331). These victories, though, are neither customary nor easily accomplished.

4.2 Planning HTML

A group of early, influential Web developers met together in early 1994 in a meeting dubbed the "Woodstock of the Web" to discuss the progression of HTML. This was before any official body had formed. "In the span of one session in one of the meeting rooms, the agenda was laid down for HTML for the next few years—how to incorporate tables, math, and the handling of graphics and photographic images" (Berners-Lee 1999: 86), none of which had not been formalized to any degree up until that point. Just a few months later, the World Wide Web Consortium (W3C) was formed, determined "to 'lead the Web to its full potential,' primarily by developing common protocols to enhance the interoperability and evolution of the Web" (94). It is in charge of all Web technologies, of which HTML is naturally a fundamental one.

4.3 W3C Activities

Like the markup language that it guides, the W3C does not cleanly fit into the mold of either a natural language or a traditional programming language standardization body; it fulfills a unique combination of roles traditionally ascribed to quite different organizations. W3C would, among other responsibilities, have a small full-time staff to develop open technical specifications, represent the power and authority of millions of developers, researchers, and users, and leverage the most recent advances in information technology" (Berners-Lee 1999: 94).

Throughout all the goings-on with browser companies making extensions to HTML, the W3C already existed. Although it seems contradictory and even a little

counter-productive, the W3C "encourages browser companies to 'innovate'—essentially, to act as test markets for new features and technologies." The rationale behind this procedure is that it allows users and committees within the W3C itself to "evaluate different solutions to a given problem and try out draft specifications in the 'real world' to see how they perform" (Meyer 1998) to determine what would work best for a final specification.

In the beginning when Netscape was the dominant browser, all the new elements it experimented with were ideal for the W3C's concept of testing and advancement of HTML. As the Browser Wars started and the motives for "innovation" became more corporate, however, this tested W3C's weight as a standards organization. Certainly the growing incompatibilities went directly against the interoperability of the Web that the W3C was meant to oversee, and the Web's "commercial success preceded the development of industry standards" (Zeldman 2003: 22).

The W3C kept a relatively low profile at this time, quietly releasing its specifications (called Recommendations, rather than standards or regulations), knowing that a marketing battle would not further the future of the Web, but that keeping up with evolving technology would. Although it admittedly had no way to require anybody or any company to abide its Recommendations, Berners-Lee maintains that the consortium, the press, and the user community all work as part of a cycle to pressure the relevant parties into complying (Berners-Lee 118). This ideology is consistent with his vision of openness that spurred the idea of the World Wide Web in the first place. Although that does finally seem to coming to fruition at present, developments generally went as would be expected at the time, which was in complete disregard to the W3C Recommendations.

4.2 Real World Activities

In reality, the simple fact was that Netscape and Microsoft were too caught up with their own concerns to pay much attention to recommendations for which there were no tangible consequences for not following. According to Zeldman, the situation for Web developers would actually have been better had the browsers ignored all Recommendations completely. As he explains it, these browsers of the late 1990s "paid lip service to some standards by supporting them partially and incorrectly. Slipshod support for standards as basic as HTML created an untenable Web-publishing environment that led in turn to unsustainable production methods" (Zeldman 2003: 25). Zeldman then offers the following analogy:

> "When a patient's appendix bursts, a qualified surgeon performs a complete appendectomy. Now imagine if, instead, a trainee were to remove half the appendix, randomly stab a few other organs, and then forget to sew the patient back up. We apologize for the unsavory imagery, but it's what Web standards support was like in old browsers: dangerously incomplete, incompetent, and hazardous to the health of the Web" (Zeldman 2003: 25).

The unsustainable production methods referenced above involve Web developers having to be familiar with which browser supported which propriety elements and having to perform time-consuming tricks to deliver to each browser the version that it could understand. In many cases, this involved creating two versions of every page on a website because there was no way for both Netscape and Explorer to agree on anything but the basics. Obviously, this was a huge waste of time, energy, money, and other valuable resources.

4.3 New Rules

The ways in which XHTML and HTML differ are relatively simple and easy to remember, as well as being very clearly defined. XHTML does not allow the kind of "tag

soup" that characterized HTML for so many years and is far closer to a programming language in terms of its lack of flexibility, which is what some standards-proponents had been hoping for for some time, as a way to force more people to create cleaner markup. That particular result is not too likely, however, since modern browsers all also have taken pains to still support tag soup, as long as it isn't incorrectly declared as XHTML. And anyway, the people brewing tag soup aren't going to be on the frontlines of the new web standards. A few of the main changes in the conversion from HTML to XHTML are below (Zeldman 2003: 156):

- For XHTML, all tags must be closed, either with a separate close tag, like `<p></p>`, or by escaping it with a space and a forward slash, like `<br />`. Not all tags in HTML had to have closing tags, and empty tags like `<br>` didn't even have a close tag as an option.
- Element names must be lowercase. While this does not violate a stated rule in HTML, which was case insensitive, the convention was to write the tag name in uppercase to be able to easily distinguish the markup from the content. XHTML is backwards compatible with older versions of HTML so that browsers that made before the introduction of XHTML will still be able to display the page.
- Attribute values must be quoted. In the examples `<img src="foo.gif" width="350" height="200" alt="Generic alt text" />`, the attributes are `src`, `width`, `height`, and `alt`, and their respective values are in quotes. In HTML, attribute values that were numbers, such as width and height did not have to be in quotes.

5.0 Conclusions

The gradual trend of HTML becoming more stringent as determined by W3C Recommendations and as implemented by major browsers is a sign of progress for the future development of the Web. For a technology becoming as globally significant as HTML is, lacking or inconsistent standards are simply not an option. In my view, what HTML has been through is a unique and fascinating linguistic and technological experiment, but practically speaking, not one that can or should intentionally continue. Its ultimate status seems to be that of strict compliance, moving it out of its own, no man's land category to be closer with conventional programming languages and distancing it from natural language.

It is worth considering, though, that HTML would not have been able to reach its current state—guided toward a known destination—without lessons gleaned from its ten-year flux. There is also the very real possibility that HTML would not have succeeded even in the beginning if it had not started very simply and unassumingly. Consequently, even though HTML may always have been meant to return to its basis as a computer language, there was no way to predict that ahead of time.

Bibliography

Abbate, Janet. *Inventing the Internet*. Cambridge: The MIT Press, 1999.

Associated Press. "French consign 'email' to trash can." 21 July 2003. *The Guardian*. 6 Dec. 2003. <http://www.guardian.co.uk/online/news/0,12597,1002708,00.html>.

Anderson, Tim. "The XML Revolution: An Interview with Charles Goldfarb." 2001. 7 Dec. 2003. <http://www.xmlhandbook.com/press/anderson.htm>.

Aronson, Larry. *HTML Manual of Style*. Emeryville: Ziff-Davis Press, 1994.

Berghel, Hall, and Douglas Blank. "The World Wide Web." In *Advances in Computing*, Vol. 48. Ed. Marvin Zelkowitz. Academic Press: New York, 1999. 178-218. As found online at <http://dangermouse.brynmawr.edu/pub/www/>.

Berners-Lee, Tim, with Mark Fischetti. *Weaving the Web: the Original Design and Ultimate Destiny of the World Wide Web by Its Inventor.* San Francisco: Harper, 1999.

Castro, Elizabeth. *HTML 4 for the World Wide Web.* Visual QuickStart Guide. 4th ed. Berkeley: Peachpit Press, 2000.

Eisenberg, David J. "How to Read W3C Specs." 28 Sept. 2001. A List Apart. 25 Nov. 2003. <http://www.alistapart.com/articles/readspec/>.

Fensel, Dieter et al. Introduction. *Spinning the Semantic Web: Bringing the World Wide Web to Its Full Potential*. Eds. Dieter Fensel et al. Cambridge: The MIT Press, 2003. 1-25.

Koch, Peter-Paul. "A history of browsers." 4 Dec. 2003. *Quirksmode*. 7 Dec. 2003. <http://www.quirksmode.org/browsers/history.html>.

"Markup language." 7 Nov. 2003 *Wikipedia*. 25 Nov. 2003. <http://en2.wikipedia.org/wiki/Markup_language>.

Meyer, Eric. "Why Browsers Haven't Standardized." 22 Sept. 1998. *Webmonkey*. 25 Nov. 2003. <http://hotwired.lycos.com/webmonkey/98/38/index1a.html>.

Open Source Initiative. 12 Nov. 2003. *Open Source Initiative*. 5 Dec. 2003. <http://www.opensource.org>.

Trask, R. L. *Historical Linguistics*. New York: Oxford University Press, 1996.

Zeldman, Jeffrey. *Designing with Web Standards*. New York: New Riders, 2003.