Dynamic HTML and Cascading Style Sheets

A Thesis

Presented to

the Chancellor's Scholars Council of

The University of North Carolina at Pembroke

In Partial Fulfillment

of the Requirements for Completion of

the Chancellor's Scholars Program

by

Sandon Jacobs

May 4, 1999

Faculty Advisor's Approval [illegible]
Date May 6, 1999

Dynamic HTML and CSS

Dynamic HTML and CSS

I. Introduction to Dynamic HTML

A. HTML: The Basics

Today, web authors face great challenges in making sites interactive. HTML's static nature limits a developer's creative choices and interactive components can be very complex to build and hard to reuse. Also, proprietary extensions make for the creation of browser-specific sites. At long last, Dynamic HTML gives us the ability to modify a Web document on the fly; i.e., HTML that's already there just not always visible. Such technology comes along to mark the beginning of the point-and-click, drag-and-drop, newly interactive Web (Methvin, "Web" 235).

Since Hypertext Markup Language, HTML, is the primary language of the Internet, a solid understanding of the basics of the language is necessary before going too deeply into the realm of Dynamic HTML. An HTML file consists of plain text together with tags, which are simply layout and formatting markers. This section will offer a description of the tags of HTML which commonly occur in DHMTL.

First are the page tags which are generally only used once per document. The first of these are the *<html>* and *</html>* tags used to define the start and end of the document. Next are the *<head>* and *</head>* tags. The information here is generally not displayed in the browser window. Inside the *<head>* tags are the tags *<title>* and *</title>*. As the name implies, this is the title of the document which appears in the browser title bar. The *<body>* and *</body>* tags contain the content to be displayed by the browser. Other tags will go inside of the *<body>* tags to change page layout.

Layout tags break the page content into logical pieces. First are header tags noted by *<h1>...</h1>* through *<h6>...</h6>*. The larger the number of the header, the smaller the corresponding text appears in the browser. Heading text is separated from the other text on the page by blank lines before and after the heading. The *<p>* tag specifies the beginning of a new paragraph and adds additional spacing to the end of the paragraph. There are other layout tags, but describing these few will be adequate for our understanding of DHTML.

Hypertext link tags identify a target URL as well as the text or image to be displayed on the page. The link can be a, Uniform Resource Locator, URL, a downloadable file for saving, or a location within the current HTML document.

A set of tags that is frequently seen in DHTML code is the *<style>...</style>* element. This tag pair is used in style sheets, both inline and external, which will be discussed in detail later. The *<style>* tags should be located in the *<head>* element of the page tags.

B. What is Dynamic HTML?

What is Dynamic HTML? DHTML, as it is often called, is and emerging technology that changes the face of Web development, incorporating interactivity and dynamic features to sites. DHTML creates relief for Web servers by shifting processing demands and scripting to the client-side, thus shortening the response time of new data. Dynamic HTML is a combination of technologies such as HTML, scripting, and object-oriented design principles all together to create individual Web pages. It is not HTML 5, a new scripting language to learn, or some new compiled language just passing through. It is a way of bringing these technologies together to form the interactive Web of the future (Heinle, "DHTML" 2).

DHTML can do things to make Web sites more navigable and easy to read, as all surfers wish. All Internet users scream for more add-ons and user-friendly features and the big two, Netscape and Microsoft, hear us loud and clear. Both have created dynamic object models and added the use of style sheets to new browsers. The intervention of the World Wide Web Consortium, W3C, is needed to work the features of DHTML into a standard for Web developers (Methvin, "Really" 237).

C. Document Object Model and the Features of Dynamic HTML

DHTML needs a way to view a document. Using static HTML, the browser views the entire document, reads the code, prints the output to the screen, and then is finished with it. An HTML document's static nature does not provide the flexibility for Web developers to create pages that look sophisticated and easily make the pages interactive. On the other hand, using DHTML, the document assumes its own unique structure called the Document Object Model, DOM for short. With the DOM, the page still exists as a text file but the browser now views it quite differently. All page elements are objects which can be easily manipulated, their attributes changed, and scripts executed on demand (Powell 62).

Tags in DHTML are accessed through the Document Object Model (DOM). The DOM describes an HTML document as a collection of objects such as images, paragraphs, and forms all the way down to the individual characters. Each object may have properties associated with it, usually in the form of HTML attributes. The World Wide Web Consortium defines the DOM as a "platform and language-neutral interface that will allow programs and scripts to dynamically access and update the content, structure and style of documents" (http://www.w3c.org/DOM). Developers can use DHTML's DOM to animate a page by moving objects around, building tree structures for site navigation, or create applications like a database front end or a game. All of this has been possible for a while to experienced JavaScript programmers, but true DHTML takes the idea further by getting right down to the actual text, styles, tags, and scripts involved (Powell 63).

The ability to change tags and attributes on the fly is the very core usage of the Document Object Model. Imagine this scenario: we have opened a site with some text and what appears to be just a list of headings. Click the mouse button on one of the headings and a paragraph of text "magically" appears underneath the heading. Pretty cool, and all of this is done with just a touch of JavaScript and a couple of style tags, both of which we will discuss in the following pages (Campbell et al 10) .

A feature known as live element positioning can turn a site into an interactive playground in which elements have mobility on the screen. Interactive formerly meant filling out forms or waiting on server-side scripts, but now it is a virtually new page almost immediately displayed on the screen. Microsoft and Netscape both bought into this idea and have tags to accomplish it. Microsoft IE uses the <DIV> tag and Netscape uses both the <DIV> and <LAYER> tags (Campbell et al 11).

Developed by Netscape, dynamic fonts are used to solve the dilemma of fonts used by developers that may not be present on the client-side. Because of this limitation, the developer does not know exactly how the page appears to the user. Font information is passed along with the site in a compressed file and is read in vector format, which is assembled on-the-fly using libraries on the client machine (Campbell et al 11).

II. Cascading Style Sheets

A. What is CSS?

CSS is a language itself, one of the many involved in the mastery of DHTML. The purpose is to create and define styles for the content of the Web page. The style can instruct such things as font size color, and face, as well as position of text and graphics in the window. Browsers have their own style sheets to interpret attributes of tags. When a CSS is encountered, the newly defined styles take precedence in conflicting cases. "Cascading" means that many styles can be used in a single Web page and the browser will follow an order or a "cascade" to interpret the information. A designer can use three types of cascading style sheets simultaneously and the browser will deliver the appropriate data in an orderly fashion; inline, embedded, and linked (Heinle, "DHTML and CSS" 1).

Unlike DHTML and the DOM, the World Wide Web Consortium has developed a standard for Cascading Style Sheets referred to as CSS. Both Microsoft Internet Explorer 3.0 and later versions as well as Netscape Navigator 4.0 and later recognize this standard.

With this type of corporate support, expect style sheets to become a big part of future Web development (Peck and Hane 46).

B. Style Control

Style control with HTML is limited. With cascading style sheets, Web developers have new and improved ways to control style issues like fonts, including their attributes such as face, size, and weight. Fonts are not the only thing though. Cascading style sheets can be used to extend basic functions to control margins, indents, and a multitude of other options (Peck and Hane 45).

Theoretically speaking, scripting languages, such as JavaScript and VBScript, have access to all properties of CSS via the DOM. What does this mean? One example is that a style applied to a link could change if the mouse is moved over it. Another is expanding text on the selection of the reader. Both topics will be described later (Heinle, "DHTML and CSS" 4).

C. Style Types

There are three primary ways to use style sheets: the inline method, the embedded method, and the external method. All three will work with DHTML, but inline and embedded are usually more common used because having all styles in a single file makes for easy viewing and modification (Campbell and Darnell 60).

1. Inline Style

The inline method is used with the tag on which it takes affect. It has no effect at all on other tags on the page, even those of the same type. ***Code Example A*** is an illustration of inline style usage. Try using this code on a Web editor and see the differences in style on different browsers (Campbell and Darnell 61).

Code Example A "inline.html"

```
<HTML>
<HEAD>
<TITLE>Example of Inline Style</TITLE>
</HEAD>
<BODY BGCOLOR="#FFCC00">

<P  style="font-size: 14pt" id="p14">
```

```
This text is enclosed in a P tag with a style.<BR>
We the people of the United States, in order to form a more perfect union, establish
justice, ensure domestic tranquility, provide for the common defense. . .
</P>

<P>
This text is enclosed in a P tag without a style.<BR>
We the people of the United States, in order to form a more perfect union, establish
justice, ensure domestic tranquility, provide for the common defense . . .
</P>

<span style="font-size: 18pt" id="span18">
This text is enclosed in a SPAN tag with a style.<BR>
We the people of the United States, in order to form a more perfect union, establish
justice, ensure domestic tranquility, provide for the common defense . . .
</P>
</span>

<span>
This text is enclosed in a SPAN tag without a style.<BR>
We the people of the United States, in order to form a more perfect union, establish
justice, ensure domestic tranquility, provide for the common defense . . .
</P>
</span>

</BODY>
</HTML>
```

2. Embedded Style

An embedded style sheet is a set of definitions inside of <STYLE> tags between the head and body sections of the HTML code. The style attributes for the entire HTML document are embedded here. ***Code Example B*** shows a style sheet embedded in an HTML document. When using DHTML and style sheets, embedded style sheets are used quite often and are easily implemented (Campbell and Darnell 64).

Code Example B "embed.html"

```
<HTML>
<HEAD>
<TITLE>Example of Embedded Style</TITLE>
</HEAD>

<STYLE>
BODY {background :#0000FF; color:#FFFF00; margin-left:0.5in; margin-right:0.5in}
H2 {font-size:18pt; color:#FF0000; background:#FFFFFF}
P {font-size:12pt; text-indent:0.5in}
</STYLE>
```

```
<BODY>

<h2 id="h2_1">Here is a red heading on a white background</h2>

The rest of the text is yellow on a blue background.<BR>

Besides the margins, this text is not indented. We the people of the United States, in
order to form a more perfect union, establish justice, ensure domestic tranquility,
provide for the common defense, promote the general welfare, and ensure the
blessings of liberty for ourselves and our posterity, do ordain and establish this
constitution for the United States of America.

<p id="p_1">
This is some indented text. We the people of the United States, in order to form a more
perfect union, establish justice, ensure domestic tranquility,  provide for the common
defense, promote the general welfare, and ensure the  blessings of liberty for ourselves
and our posterity, do ordain and establish this constitution for the United States of
America.

</BODY>
</HTML>
```

This method works well for styles used only in one document. When uniformity across sites is needed, linked style sheets may be the more logical choice.

3. Linked Style

Linked or external style sheets can be used on multiple HTML documents. The style sheet itself is an independent file on its own. This is why it is called linked, because the "style-file" is linked to the HTML files that use the defined styles. Styles are defined inside the <STYLE> tags and the "style-file" is given a new extension of ".css". ***Code Example C*** is a simple external style sheet much like the style we saw in the embedded example earlier. ***Code Example D*** is an HTML document using the styles in ***Code Example C*** (Campbell and Darnell 65).

Code Example C "style_file.css"

```
<STYLE>

BODY {background :#0000FF; color:#FFFF00; margin-left:0.5in; margin-right:0.5in}
H2 {font-size:18pt; color:#FF0000; background:#FFFFFF}
P {font-size:12pt; text-indent:0.5in}

</STYLE>
```

Code Example D "linked.html"

```
<HTML>
<HEAD>
<TITLE>Example of External Style</TITLE>
<link rel=stylesheet href="style_file.css" type="text/css">
<!–The above line is a reference to the file that defines the styles used in the page.
    This is like an "include" statement in a C++ program.–>
</HEAD>

<BODY>
<h2>Here's a heading</h2>

<p>
We the people of the United States, in order to form a more perfect union, establish
justice, ensure domestic tranquility, provide for the common defense, promote the
general welfare, and ensure the  blessings of liberty for ourselves and our posterity, do
ordain and establish this constitution for the United States of America.
</p>

</BODY>
</HTML>
```

There are instances in which different style sheets can be used together. In such cases, there exists also a hierarchy for the sheets. Linked style sheets are applied globally to the document. If there is an embedded sheet on the same page with a linked sheet, the embedded sheet overrides the linked sheet in cases of style conflict. Inline styles override both embedded and linked styles (Campbell and Darnell 68).

D. Organizing Style

There are two basic ways to organize style sheets, thus making them much easier to understand and use. Grouping style sheets reduces the amount of typing by placing styles into logical groups. Classes allow variations on a style and also enable more than one style to be used on the same HTML tag (Campbell and Darnell 80).

Grouping saves typing by putting like styles together. For instance, if the same style attributes are given to three different tags, the style can be defined and typed once with the attributes assigned to all corresponding styles, like this (Campbell and Darnell 81):

```
H1, H2, H3 {font-family: arial; font-size 14pt; color: #ffcc00}
```

We can also do this with margins. The top, right, and left margins are defined in one line like this:

```
BODY {margin: .10in .75in .75in}
```

New variations on style are possible through the use of classes in style organization. An extension can be added to any HTML tag to assign a class to it. The name can be almost anything the programmer desires. A simple example is used with the <P> tag. Once the class is defined, an additional attribute can activate the class when the <P> tag is used. Here is an example (Campbell and Darnell 83):

```
<STYLE>
<!--
P {font: arial 12pt/11pt normal}
P.left {text-align: left}
P.right {text-align: right}
-->
</STYLE>

<P class="left">
```

This will justify the text to the left-hand margin.

<u>III. Other Style Changers</u>

A. JavaScript Style Sheets

JavaScript Style Sheets (JSSS) are an innovation of the Netscape Corporation. This is a completely different way of defining styles in an HTML document. JSSS allows the style sheet to ask the browser about information such as the operating system, size of window, color compatibility, and other characteristics before assigning values to the tags and classes, allowing the page to configure itself to the environment of the client's machine (Musciano 1).

JavaScript is a lightweight interpreted programming language with fundamental object-oriented capabilities. The core of the language, for general purposes, has been embedded in the source code of Netscape Navigator and other browsers and padded for web programming with the addition of objects representing the browser window and the contents of the window. For this reason, JavaScript is often referred to as "client-side" and the "executable code" can be embedded directly in the HTML document, making the page no longer static, but dynamic in nature (Flanagan 1).

The syntax of JavaScript greatly resembles that of C, C++, and Java, with certain constructs from the languages being present. Variables in JavaScript carry no type specifications, making JavaScirpt objects more like Perl arrays than the objects of the aforementioned OO languages. JavaScript is not compiled like C and C++, or compiled into byte-code like Java, but interpreted at its execution time (Flanagan 3).

As with any innovation in technology, there are some common misconceptions about JavaScript. JavaScript is not Java simplified. It is not even a product of Sun Microsystems as is Java. The similarity of the names is just a marketing strategy to get the world to trust the familiar name. The language was originally named LiveScript, and changed to JavaScript at the last minute. Most people also feel that a scripting language is much simpler than a programming language. Thus, non-programmers often feel they can just jump into JavaScript, which appears at first glance to be very simple. As one digs deeper into the functionality of the language, the complexities become very clear. So, the question is now what can JavaScript do? The power of the language lies within the browser and the document-based objects JavaScript supports (Flanagan 10).

JavaScript can control document appearance and content. An example of this is the *write()* method of the Document object that allows HTML to be dynamically printed to the browser window. This gives the programmer great flexibility of content based on the browser and platform to print constantly changing content such as time, date, or golf scores. Page attributes can also be defined by a JavaScript object. This technique is very useful in programming for frames and the dynamic generation of those frame contents can often replace traditional CGI scripts (Flanagan 13).

Some JavaScript objects hold control over the browser and its behavior. The Window object allows for pop up dialog boxes for simple messages. This object also supports the use of multiple browser windows. JavaScript does not allow direct creation and manipulation of frames within a window, but the HTML tags to create the desired frame layout can be dynamically generated. The Location object provides for the download and display of a URL in any window or frame of the browser. The "revisiting" of previously seen sites is made possible by the History object (Flanagan 12).

JavaScript's Document object lets the code interact with the document content. A simple application such as a calculator can easily be implemented in JavaScript and run on the client side. JavaScripts are also used with HTML forms. The JavaScript is used to check for errors in user input on the client-side. This eliminates the wait of a round trip to and from the server. This also allows for the CGI script that uses the form input to be a much simpler code, because it does not have to perform any error checking functions. In many cases, JavaScripts and CGI scripts do work very well together (Flanagan 7).

Among interactions with the user, the most popular are event handlers. These pieces of code are executed when the corresponding event occurs, such as a mouse-click. Event handlers are crucial to a graphical interface that requires an event driven model. JavaScript can initiate these actions in response to an event. Examples of such events are status lines and pop up boxes (Flanagan 22).

Netscape 4.0 introduced three new objects to changing styles. They are *tags*, *classes*, and *id*. Tags are the mark-up elements of an HTML document. As children of the Document object, they are accessed using the "dot-notation" syntax document.tags.*tagname,* where *tagname* is an HTML tag like *H1*, *STRONG*, or *P*. Classes are another division of tags, like the *left* and *right* versions of the *P* tag explained earlier. These classes are not to be confused with Java classes. They are also children of the Document object, with a syntax of document.classes.*classname[.tagname]* where *classname* is the identifier of the class and *tagname* is optional for application to a specific HTML tag; it can be set to *all* to apply the class to any tag on the page. The *id* is a unique identifier to a particular tag. It is applied like so: <H1 id="SAM">. This object is now identified as "SAM." The syntax of this member of the Document object is document.ids.*idname* where *idname* is the name in the *id* attribute. The previous example would read document.ids.SAM, because JavaScript is case sensitive (Musciano 3).

B. HTML Replacement

HTML Replacement is a feature of DHTML that allows the developer to change text and graphics associated with a specific HTML tag or tag pair. Tags like the <P> tag and the <DIV> tag are useful when replacing whole sections of tags at a time. The following examples will use the <DIV> tag. A common way to use HTML replacement is by setting up a paragraph that appears on a site after a mouse click or other interaction. A page written with Dynamic HTML can appear load in one format and, after responding to a user interaction, dynamically change the content of the site, thus dynamically changing the tags without the long trip back to the server. Similarly, replacement for graphics is possible so that a graphic changes to another graphic or to text that appears to be "behind" the image. HTML replacement can be thought of as having control over multiple sites within one site that changes by the features of DHTML and not through command of a server (Campbell and Darnell 238).

So why do we need to change text content? Text content changes take style to a whole new level. The motives for it are similar to that of style changes: retain the audience and provide interaction with the user. This is quite like the interaction of dialog boxes, message boxes, and status bar comments found in traditional GUI applications (Campbell and Darnell 238).

The cornerstone of HTML replacement is DHTML's method of defining existing HTML tags and text in a block as an object that can change dynamically. Any and every HTML tag pair that can hold text can be used in HTML replacement. Instantiate the elements to be changed as objects using the *ID=* attribute. That gives for text element properties to dynamically change the contents within the tag pair. The properties are innerHTML, outerHTML, innerText, and outerText. To change the HTML element, assign a new string to the appropriate property of the element. Use the following descriptions as a guide:

innerHTML: defines the contents of an element object with HTML tags and maintains the existing tags that define the element.

outerHTML: defines the contents of an element object with HTML tags, but replaces the element itself with the outer tags in the assignment string.

innerText: defines the contents of an element, but evaluates the assignment string as text, not HTML.

outerText: defines the new contents of an element and eliminates the element tags at the same time, evaluating the new as text, not HTML.

These four properties can be changed for any and every valid text-containing HTML tag pair. A DHTML event could trigger the script that makes the content change (Campbell and Darnell 241).

Graphics too can be dynamically altered. Why change graphics? Well, WHY NOT? The dynamic changing of graphics allows the world of Web development to keep pace with the software applications market. Without this, there are no online games, animations, or the popular online movies. I do not think movies will ever be feasible without the use of an extra file, but that's not the issue here (Campbell and Darnell 247).

<u>IV. Dynamic Design</u>

A. Designing a Dynamic Site

There exists a number of technologies that can be integrated into a Dynamic HTML site. Among these a Java, VBScript, and ActiveX, to name a few. No matter how interactive the features of the site may be, there are certain design issues to be addressed and decisions to be made. Many of these decisions are the same for static and dynamic sites alike (Campbell and Darnell 366):

- How busy should the page or pages be?
- Should I use frames?
- Should I make the pages long or short and how many pages?

- What browsers do I use for testing?
- How can I make my next site in less time?
- How do I attract an audience to my site?

B. How much is too much?

The standards for HTML only cover syntax and appropriate use of HTML and set no limits on how much of the language to use in a page. Browsers can set limits. Netscape, for example, only supports layers up to 10 deep. Other than the browser, the developer is the decision maker on the functionality of the Web page. Everyone runs across the site that tries to do a little too much (you know, the one with 3 or 4 Java applets running at the same time while trying to load 25 images). That is usually too much information for the average Web surfer's machine.

If highly dynamic is the preference of the developer, consider giving the audience an option for a text-only version or a static option. A page that is too dynamic is one that takes the attention of the surfer from the data presented to the moving stuff on the page. When developing, the most important thing is to keep the audience in mind; the site is for them. When adding that exciting new feature, make sure that it does not make for one too many (Campbell and Darnell 372).

IV. References

Campbell, Bruce and Rick Darnell. Teach Yourself Dynamic HTML in a Week. Sams.net Publishing, 1997.

Flanagan, David. JavaScript: The Definitive Guide, Second Edition. O'Reilly & Associates, Inc, 1997.

Heinle, Nick. "Dynamic HTML." WebReiview.com. http://webreview.com.

Heinle, Nick. "Dynamic HTML and Cascading Style Sheets." DHTML Zone. http://www.dhtmlzone.com/articles/dhtmlcss.html.

Methvin, David W. "Really Useful DHTML". Windows Magazine, Aug98, Vol. 9 Issue 8, p237, 2p, 3c.

Methvin, David W. "Web pages with style". Windows Magazine, May98, Vol. 9 Issue 5, p235, 2p, 1 chart, 3c.

Musciano, Chuck. "The dynamic, powerful abilities of JavaScript Style Sheets." SunWorld-Webmaster, April 1997 (http://www.sunworld.com/swol-04-1997/swol-04-webmaster.html).

Powell, Thomas A. "The power of the DOM." Internetweek, 09/29/97 Issue 683, p61, 3p, 2 charts, 1 diagram.

Peck, Robin and Paula J. Hane. "Web page design standards: Part I." Information Today, Oct98, Vol. 15 Issue 9, p45, 2p.