Pacific University
CommonKnowledge

Volume 6 (2006)

Interface: The Journal of Education, Community and Values

9-1-2006

Typographical Control via Cascading Style Sheets

Michael Geraci

Typographical Control via Cascading Style Sheets

Typographical Control via Cascading Style Sheets

Posted on **October 1, 2006** by **Editor**

By Michael Geraci

Part One: The need for improved typography
Part Two: What the research says about typographical improvements
Part Three: Typographical Control via Cascading Style Sheets

If you've been tuning in to this space for our previous installments, I hope that I've whetted your appetite for improving the quality of your on-screen type. The good news here is that the Cascading Style Sheet controls for type are a very small sub-set of the CSS "language", and thus present a manageable undertaking for even the most techno-challenged. Using CSS to establish good typography is also an excellent way to dip your toe in the CSS waters. Once you understand the basic syntax and structure of the language, you'll be much more likely to expand your usage into other realms like color, imagery, and page layout.

This article does not attempt to be a complete resource to the CSS language, there are myriad sites that provide in-depth coverage and tutorials for this. I have provided links to some of my favorites at the conclusion.

CSS in a nutshell

CSS is not really a language per se, it is really a set of rules for how information should be presented. It's more like grammar in that respect. The beauty of CSS is that the rules are largely user-defined (in this scenario, you, oh dutiful Web designer, are the "user") as opposed to pre-defined. The presentational rules of CSS are contained in one or more style sheets that either live inside the HTML markup of your pages or they are contained in an external document and then linked up to your HTML pages. The beauty of the latter system is that a single CSS source can control the presentation of all of your Web pages and therefore you can make sweeping changes across an entire site (or even a collection of sites) by modifying just the CSS document(s). A truly

remarkable example of this is the CSS Zen Garden web site where end-users are able to submit their own CSS rendition of a common page of HTML markup. At the time of this writing, there are 979 different designs for this page that range from subtle elegance to awe-inspiring eye-candy.

The power and magic of this system relies on one simple rule: that HTML markup must be well-structured; that is, our paragraphs must be enclosed in the appropriate HTML tags along with our headlines, blockquotes, lists, and links, etc.. Gone are the days where you could slip in a <dt> tag to create indented paragraphs. Under the new rules, only true Definition Terms get to be set in the <dt> element. Also taboo are tags that had nothing to do with structure, like our old friend the <font> tag. We still get to specify typefaces in the new system, the difference is that we attach typeface and other presentational styles to the structural markup. For example:

```
p { font-family: Verdana; }
```

Here we see the very basic syntax of CSS. It starts with a "selector" in this case, it's the "p" which references the (paragraph) element in HTML. The selector is followed by a set of curly brackets that contain the property that we want to set, here it's typeface via the "font-family" property. "Font-family" is a pre-defined item in CSS, and it's these "keyword" items that one must learn in order to become fluent in CSS. The hyphenated format is very standard in CSS, so any property that involves multiple terms is hyphenated, "font-size", "font-style", and "text-align" all demonstrate this. Finally, there must be a value applied to every property. In this case, the typeface Verdana is being applied to the font-family property. *Don't forget to separate properties and values with a colon(:) which should directly succeed the property keyword.*

What this rule achieves is the display of the Verdana typeface for all text that is set in paragraph tags: … . We'll get to more rules in a bit, but first we must find a home for our CSS.

Where to put CSS

As I stated before, CSS can either live inside your HTML markup (we call this an embedded style sheet) or in an external style sheet that is linked to our HTML file. For simplicity's sake, we're going to start out with embedded style rules. Our next installment will address external style sheets.

All HTML files require a <head> element. That all-important section of the markup where meta-information lives. Typically the document's <head> will contain the title of the page, keywords, a short description, and now your style rules. Yes, embedded CSS rules are added to the <head> of your document. To manually add style sheet rules then, you simply add the following element anywhere within the <head> element of your page:

```
<style type="text/css">
```

```
</style>
```

All of your style rules will be added between those two tags. Let's start by setting the default typeface for the entire page. This will be done by applying a rule to the <body> of our page. Unlike the <head> of the page, the <body> represents the portion of our Web page that renders in the browser for people to see. By setting a default typeface for the body, we are establishing some page-wide consistency. The following would be added inside our <style> element.

```
body, td {font-family: verdana, arial, "Trebuchet MS", sans-serif;}
```

What I've done here is picked two different selectors, the body and table data, and I've set them all to Verdana. The other typefaces that follow are the back-up plans in case Verdana is not available, the browser simply moves to the next typeface until it has a match installed on the user's system. Two things are worth noting here: 1) Because Trebuchet MS contains a space in its name, it should be enclosed in quotes in order to be treated as a single value; and 2) "sans-serif" is not the name of a typeface it is a generic class, which serves as a last resort that tells the Web browser, "If you've come this far without a match, I don't care what you use as long as it is a sans-serif typeface." Finally, the listing of values ends with a semi-colon, which officially ends the declaration.

By using "body" as the selector in the example above, I've established a default font for all elements on the page that involve the display of text. So this one rule will trickle down to include all: paragraphs, lists, headings, blockquotes, etc. The "td" was thrown into the selector mix to accommodate older Web browsers that treated text inside of table cells differently from the rest of the page text.

Taking it one step further, let's say that we want our HTML Headings to use a different typeface. Here's a new rule for our style sheet that sets the typeface of headings:

```
<style type="text/css">
body, td {font-family: Verdana, Arial, "Trebuchet MS", sans-serif;}
h1, h2, h3, h4, h5, h6 {font-family: "Lucida Grande", "Gill Sans", Arial, sans-serif;}
</style>
```

Typographical quick hits

It's one thing to set your typeface, but what about all those other typographical controls mentioned in the previous article? Now that you understand the basic mechanics of CSS, adding any of the following should be easy.

PROPERTY	SETS	EXAMPLE
font-size	font size	body {font-size: 12px;}

color	type color	body {color: #224455;}
text-align	justification	body {text-align: left;}
text-indent	first line indent	body {text-indent: .5cm;}
line-height	paragraph leading	body {line-height: 1.5;}
width	text container width	body {width: 500px;}

There's a few things in the above list that need highlighting. First, notice that CSS allows for many different units of measurement. You can mix and match to your hearts desire, just be sure to use the correct abbreviation for the unit and enter it directly after the value that it appends, no spaces. Type size can be set using pixels (px), points (pt), centimeters (cm), millimeters (mm), inches (in), and many others. Colors can be specified as hexadecimal values, as seen above. These are designated with the pound (#) sign. If you don't know how to select hex colors, then you can always specify colors by name, but browser support for all but the most common color names is spotty and therefore, may lead to some unexpected results. For the graphically inclined, CSS allows unprecedented color selection by allowing RGB color values to be specified in the following syntax:

```
h1 {color: rgb(255, 100, 0);}
```

This rule sets all <h1> headings to a medium orange by combining the full value of red (255) with a medium value of green (100) and no blue (0).

To reiterate, it is a simple matter of listing all the selectors that you want to declare a property for in a comma-separated format. So to color all of my HTML headings in this same color, we would do this:

```
h1, h2, h3, h4, h5, h6 {color: #FF6400;}
```

That's the same orange used before, but expressed in hexadecimal form.

Rules; rules; rules;

So far, we've been adding single declarations to our CSS rules. What if there is more than one property that you want to set for your page? Look at the following example:

```
body, td {
font-family: verdana, arial, sans-serif;
font-size: 12px;
color: #000033;
width: 500px;
```

```
line-height: 1.5;
text-indent: 20px;
}
```

This is just one CSS rule that establishes some page-wide typographic styles. First, you probably noticed that I put each declaration on its own line. This is a standard convention that simply makes the rule easier to read. Computers of course read it in as one continuous stream of characters and do not care how we format it, just as long as there's a matched pair of curly brackets and semi-colons after each declaration. Additional rules would simply begin on the next available line like this:

```
h1, h2, h3 {
font-family: "Lucida Grande", arial, sans-serif;
color: rgb(100, 50, 50);
font-weight: bold;
}
h4, h5, h6 {
font-family: verdana, sans-serif;
color: #990000;
font-weight: normal;
font-style: italic;
}
```

As you can see, there are a number of type specific properties that let you control the presentation of your Web-based text. Remembering them all along with their possible values is definitely a challenge. Instead of trying to memorize all of them, find yourself a good reference, either online or in print, and keep it handy when your editing your Web content. I like all of Eric A. Meyer's CSS guides in print format and the handy online reference at w3schools.

Next up

In the next installment, we'll take a look at applying CSS in Adobe/Macromedia Dreamweaver.

Learn more about CSS:

University of Minnesota at Duluth: CSS Workshop
WestCiv, Web Standards Software and Learning
Create Web Magic: CSS101
W3Schools: CSS Tutorial
Sitepoint: The CSS Anthology

This entry was posted in Uncategorized by **Editor**. Bookmark the **permalink [http://bcis.pacificu.edu/interface/?p=3286]** .