Paper 014-2009

CSSSTYLE: Stylish Output with ODS and SAS® 9.2

Cynthia L. Zender, SAS Institute Inc., Cary, NC

ABSTRACT

It has become the standard for most company Web sites to use cascading style sheets (CSS) to standardize Web site look and feel. Now, with ODS and SAS® 9.2, you can design a single CSS file to be used with ODS RTF, PDF, and HTML. In the past, CSS usage was limited to files created with ODS HTML.

This paper provides an introduction into the use of the new CSSSTYLE option in SAS 9.2. This option allows you to use cascading style sheet (CSS) style specifications for RTF and PDF files, in addition to HTML files. This paper will include a brief introduction to CSS syntax and some of the features, such as @media CSS sections, that are particularly useful when creating ODS output. Then, all the methods of using CSS with ODS are discussed, with an emphasis on the new CSSSTYLE option. In addition to these topics, a job aid will be provided that outlines the most commonly used CSS properties and property values.

INTRODUCTION

Hypertext Markup Language (HTML) , in one form or another, is the lingua franca of the World Wide Web. In the beginning, Web pages mixed embedded style information with content. The content of ODS-created Web pages is tabular data – created by SAS® procedures or processes. For many reasons, mixing Web style information with Web content is not a flexible way to deliver content. Some people visit Web sites with iPhones, Palm devices, or Blackberries and not all of those devices use "regular" HTML for display of the content. Some people want content delivered in PDF or RTF form. Other people use applications that treat the HTML page as data to load into other applications. For any of these content delivery methods to work smoothly, it makes a developer's job easier if the style information is separate from the data content.

The initial implementations of HTML, HTML 2.0, and HTML 3.2 allowed style information to be embedded in the HTML tags like this HTML used for a Header cell:

```
<TD ALIGN=LEFT bgcolor="#6495ED">
<font face="Arial, Helvetica, sans-serif" size="2" color="#FFFFFF">
<b>Name</b></font></TD>
```

But, in a large table, embedded style information for **every** table cell caused HTML files to load slowly and take up more space on the server than necessary. With the HTML 4.01 specification, the W3C (World Wide Web Consortium) adopted the standard of CSS usage for style information. After all, why repeat the style specifications (like that above) for every Header cell when you could provide the style property specifications only one time and then refer to that set of style properties for as many Header cells as were in your report?

For example, in an HTML 4.01 HTML file, there could be either an in-line style section or a reference to an external cascading style sheet. In either implementation, the HTML for a Header cell would look considerably different from the tag above:

```
<th class="l Header" scope="col">Name</th>
```

In order for this class selector reference to work, the CSS information used with this Web page must have a class selector that contains all the style properties needed by the browser to render the HTML page. The TH tag uses a class selector named 'L' (for left alignment) and another class selector named 'Header', as shown below:

```
.l {text-align: left }
.Header
{  font-family: Arial, Helvetica, sans-serif;
   font-size: x-small;
   font-weight: bold;
   font-style: normal;
   color: #FFFFFF;
   background-color: #6495ED;
}
```

No matter which method you use to set the style properties of the Header cell for the Name variable, the Header cell looks the same in the browser. The header cell is shown with three data cells for comparison purposes:

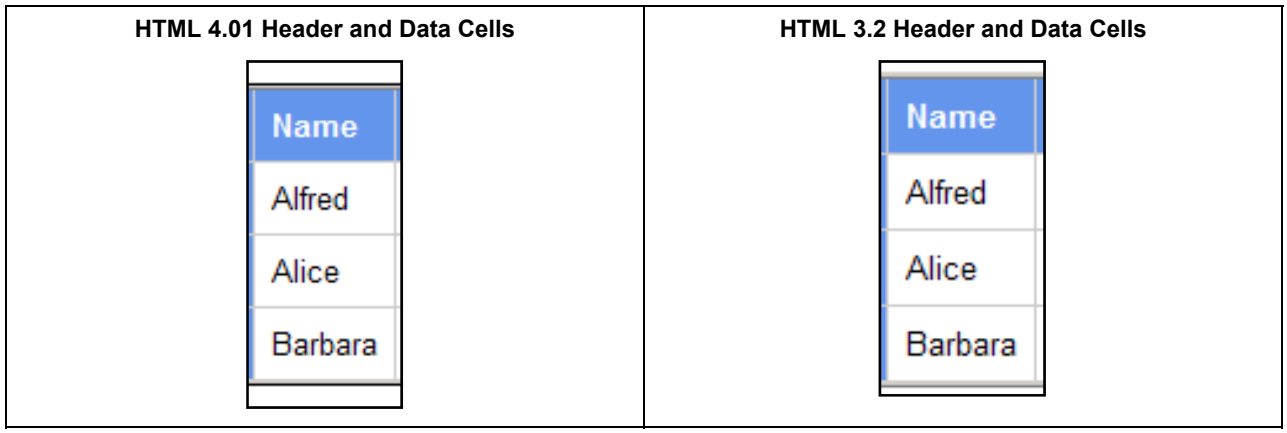

Figure 1: Output Created with HTML 4.01 method versus HTML 3.2 method

OVERVIEW OF CASCADING STYLE SHEET SYNTAX

A comprehensive study of CSS syntax falls outside the scope of this paper. ODS doesn't use the full spectrum of CSS capabilities (such as absolute positioning), mostly it uses CLASS selectors, HTML type selectors, and with the CSSSTYLE= option, allows the use of an @MEDIA rule from within a CSS file for ODS output.

Basic CSS syntax is a very simple syntax, as you can see above. If you compare the style properties within the curly braces above to the embedded version of the style properties, you can find everything from the embedded method in the CSS method, as shown in Figure 2. (The figure is both color coded and numbered for your convenience.)

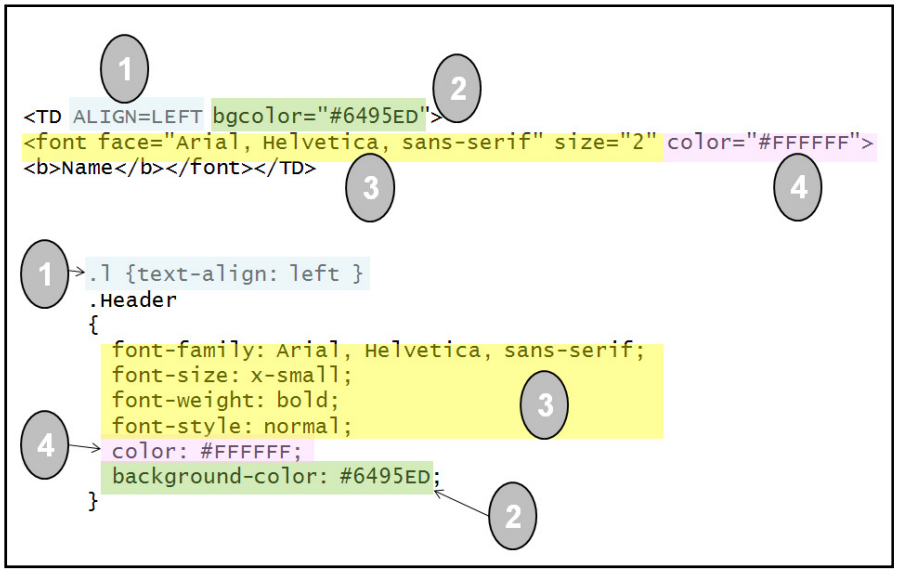

Figure 2: Comparison of Embedded Method with CSS Method of Specifying Style

The style property names that correspond to the embedded style attributes are very straightforward and easy to understand. For example "ALIGN=LEFT" (#1) can be easily spotted in the "text-align: left" style property. Background color (#2) is spelled out as "background-color" in the CSS method instead of "bgcolor" in the embedded method. To my eye, the font specifications (#3) are easier to follow in the CSS method, than in the embedded method. And, the "COLOR" specification (#4) is the same, except for the equal sign.

The ability to use this property pair syntax is one of the hallmarks of CSS syntax. It is one of the things that differentiates a CSS style specification (bottom example) from an HTML tag attribute (top example). For example, you can use HTML type selectors in your CSS specification, so a <TD> tag would cause the browser to use an HTML type selector, if there was one in the CSS file However, CLASS selectors identify, for the browser, which "package" of style properties should be used when the CLASS= HTML attribute is set to that selector. In this case, CLASS="Header" in the HTML file would cause the browser to look for a .Header specification .

Inside the curly braces used with the CSS method are CSS property pairs. You might notice that the property pairs have a very regular form: *property:value.* Technically, everything within the CSS curly braces is known as a "declaration." The rules for these style declarations are:

- Declarations consist of property pairs.
- Declarations are enclosed in { }.
- Each property pair has a colon that separates the property and its value.
- Multiple property pairs are separated by semicolons.

If you want to know more about CSS, you have to decide which of the CSS specifications to investigate. There are actually three CSS specifications: CSS 1, CSS 2, and CSS 3. Most browsers support CSS 1 and CSS 2 level specifications. You will probably want to consult your company Webmaster, if your reports and CSS files will live on the Web, because your Webmaster might only allow a certain level of CSS on the company Web site.

Also, some software vendors write *extensions* to the CSS specification, so they can implement styles in a fashion that is tailored to their software. For example, Microsoft Office has a whole set of "mso-" style properties that are designed to work well with Microsoft Office 2000 and higher.

If you do use cascading style specifications for your HTML pages, as originally designed, there are two methods for specifying CSS information with an HTML page:

- Specify CSS information in an external file and point to the external file with a <LINK> tag. (or use the CSS2 @IMPORT rule to import multiple CSS files into one CSS style section)
- Place CSS information as an in-line <STYLE> section inside the HTML file. (This is the ODS HTML default behavior in SAS 9.)

The Output Delivery System supports either method through the use of the STYLESHEET= option with ODS HTML and other ODS Markup HTML-related destinations (such as HTML3, HTMLCSS, and so on). In the next section, the original method of specifying CSS files with ODS HTML will be explored.

CSS AND ODS HTML: THE ORIGINAL METHOD

The ODS HTML destination treated style differently in SAS 8 versus SAS 9. With SAS Version 8, you got HTML 3.2 with embedded style information by default. With SAS 9 versions (before 9.2), you got HTML 4.0 tags with in-line style information by default. The original method for explicitly creating and using CSS files was

1. To create an external CSS file and build an HTML <LINK> tag that pointed to the new CSS file you use this code, with only the STYLESHEET= option. If the CSS file that is named already exists in the output path location, then it will be overwritten:

```
ods html path='c:\temp' (url=none)
         file='make_css.html'
         style=sasweb
         stylesheet='sasweb_demo.css';
  proc print data=sashelp.class(obs=3);
    title '1) Creating and Using a CSS file';
  run;
ods _all_ close;
```

2. To reference an already-existing CSS file (such as that created in the code above) and build an HTML <LINK> tag that points to the existing CSS file, you use this code, with the URL= suboption:

```
ods html path='c:\temp' (url=none)
         file='use_css.html'
         stylesheet=(url='sasweb_demo.css');
  proc print data=sashelp.class(obs=3);
    title '2) Using an Existing CSS file';
  run;
ods _all_ close;
```

3. Place CSS information as an in-line <STYLE> section inside the HTML file. This is the default ODS HTML behavior, so there's no need to demo this method of using CSS.

The output from both programs is the same. And, the <LINK> tag is essentially the same, of course if we had used a different CSS filename in #2 method, then that <LINK> tag would be different. Figure 3 shows the output and the generated <LINK> tags. Note that in the method 2 code, we no longer needed to reference STYLE=SASWEB because the CSS file provided all the style property pairs needed by ODS.

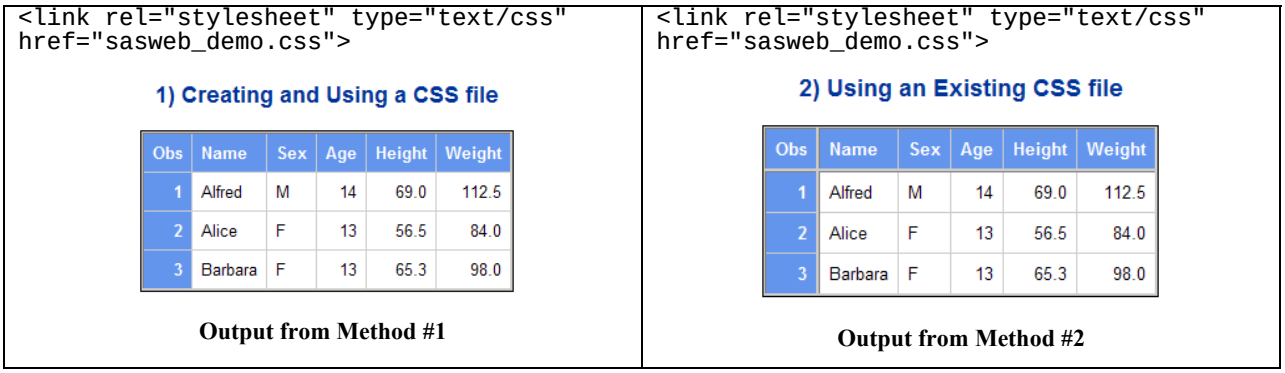

Figure 3: Comparison of Embedded Method with CSS Method of Specifying Style

Normally, if you create a CSS file using method #1, you use it as the basis for making changes to the property pairs, because it has the names of all the CLASS selectors that ODS is expecting to find in a CSS file. Then, once you have a changed CSS file, you would use method #2 to use the changed CSS file.

Now, with SAS 9.2, the CSSSTYLE= option represents an alternative approach to the use of CSS style sheet files, not just with ODS HTML, but with ODS RTF and ODS PDF as well.

CSSSTYLE: THE NEW METHOD

The simplest method for invoking the CSSSTYLE= option is almost exactly like the **STYLESHEET= option:**

```
ods pdf file='c:\temp\demo3_cssstyle.pdf'
        cssstyle='c:\temp\sasweb_demo.css';
ods rtf file='c:\temp\demo3_cssstyle.rtf'
        cssstyle='c:\temp\sasweb_demo.css';
ods html path='c:\temp' (url=none)
        file='demo3_cssstyle.html'
        cssstyle='sasweb_demo.css';
  proc print data=sashelp.class(obs=3);
    title 'Using the CSSSTYLE Option';
  run;
ods _all_ close;
```

The reason that ODS HTML uses the PATH= option is the same reason here as in the STYLESHEET= option. ODS needs the location for the CSS file. If you do not have a PATH= option, ODS looks in the current working directory, which might or might not be the location of the CSS file. Because ODS PDF does not have the PATH= option, the CSSSTYLE= option uses a fully qualified path and file location for the CSS file. ODS RTF could use either method – the PATH= option, like ODS HTML or the fully qualified location, like ODS PDF. If you download the demo programs for this paper, you will see that the DEMO3_SIMPLE_CSSSTYLE.SAS program file contains a bonus section that shows the alternate RTF invocation with the PATH= option. All three outputs from the above program are shown in Figure 4.

If you look in the HTML file, you will see an in-line <STYLE> section instead of a <LINK> tag. If you look in the RTF file, you not find any reference to the CSSSTYLE option value. That's because the CSS information is used appropriately by the destination. Essentially ODS translates from CSS property pairs to the type of style reference that is required by the destination. Because the PDF file is in a proprietary format, we can't look inside the PDF file, but we can infer, from the use of the SASWEB style for the output, that the CSS information was also treated appropriately for Portable Document Format files.

What is exciting about the new CSSSTYLE= option is the fact that it enables you to use the SAS® Enterprise Guide® style editor to make changes to a SAS style definition. Then, you could use the CSSSTYLE= option for HTML, RTF, and PDF in order to use the changed CSS file with ODS. With the newest release of SAS Enterprise Guide (4.2), you would have to use a SAS Enterprise Guide code node to be able to specify the CSSSTYLE= option. However, it is expected that upcoming releases of SAS Enterprise Guide will have a menu method for specifying CSSSTYLE for HTML, RTF, and PDF destinations.

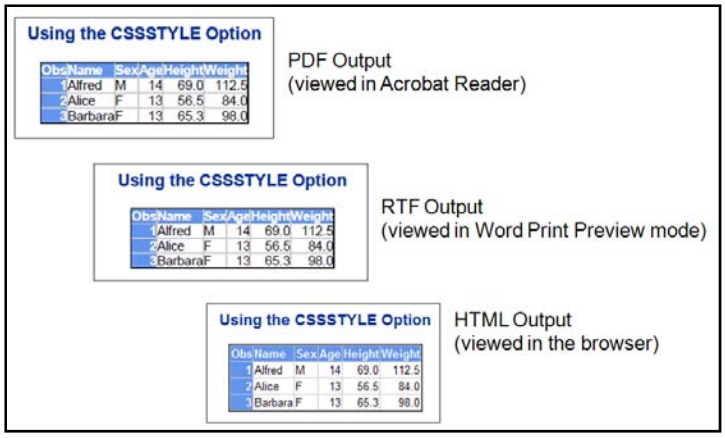

Figure 4: Output from Simple CSSSTYLE= Option Usage

However, the most exciting feature of the CSSSTYLE= option is the ability to reference CSS @ MEDIA rules in your CSSSTYLE option invocation.

ADVANCED CSS TOPICS: THE @MEDIA RULE

Before showing the final examples of using the CSSSTYLE= option, it is necessary to take a step back into the CSS world to talk about the @MEDIA rules of the CSS specification and how they are meant to be used.

Nobody explains the @MEDIA rule capability better than the group that wrote the specification. If you refer to this W3C Web site about CSS, www.w3.org/TR/CSS2/media.html, it explains that:

> "One of the most important features of style sheets is that they specify how a document is to be presented on different media: on the screen, on paper, with a speech synthesizer, with a braille device, and so on.
>
> Certain CSS properties are designed only for certain media (for example, the 'cue-before' property for aural user agents). On occasion, however, style sheets for different media types might share a property, but require different values for that property. For example, the 'font-size' property is useful both for screen and print media. However, the two media are different enough to require different values for the common property; a document will typically need a larger font on a computer screen than on paper. Experience also shows that sans-serif fonts are easier to read on screen, while fonts with serifs are easier to read on paper. For these reasons, it is necessary to express that a style sheet -- or a section of a style sheet -- applies to certain media types. "

In the CSS specification, the valid media types belong to media groups. We're going to concentrate on illustrating the use of the @MEDIA rule using the PRINT and SCREEN media types. The list of media types defined in the CSS 2 specification are: all, aural, Braille, embossed, handheld, print, projection, screen, TTY, and TV. The @MEDIA type names are not case sensitive in their specification.

Of necessity, the CSS file used for the next examples will be kept simple and will contain minimal style changes. Your CSS files can be as complicated as you need them to be. The purpose here is to show how the @MEDIA rule can be used to your advantage. For this purpose, the "print" section of the @MEDIA rule sets a gray and white color scheme with a serif font, like Courier; and the "screen" section of the @MEDIA rule sets a more colorful pink and purple color scheme with a sans-serif font, like Arial.

Let's look at a CSS file (DEMO4.CSS) that contains @MEDIA rules:

```
.body
  {background-color: white;
   color: black;
   font-family: serif}

.header, .rowheader, .footer, .rowfooter, .data
  {border: 1px black solid;
   color: black;
   margin-left: 2pt
   margin-right: 2pt
   padding: 5px;
   font-family: serif}

.header, .rowheader, .footer, .rowfooter
  {background-color: #eeeeee;}

.table
```

```
  {background-color: #eeeeee;
   padding: 7px
   border-spacing: 0;
   border: 1px black solid;}

.proctitle
  {font-family: Arial, Helvetica, sans-serif;
   font-size: large;
   font-weight: normal;
   color:green;}

.systemtitle
  {font-family: Arial, Helvetica, sans-serif;
   font-size: large;
   font-weight: bold;
   color: purple;}

@media screen {
.header, .rowheader, .footer, .rowfooter
  {color: white;
   font-weight: bold;
   font-size: medium;
   font-family: Arial, Helvetica, sans-serif;
   background-color: purple;}

.data
  {font-size: small;
   font-family: Arial, Helvetica, sans-serif;
   background-color: #ffccff;}
}

@media print {
.header, .rowheader, .footer, .rowfooter
  {color: black;
   font-size: 11pt;
   font-family: Courier New, Courier, serif;
   background-color: #aaaaaa;
   padding: 5px;}

.data
  {font-size: 9pt;
   font-family: Courier New, Courier, serif;
   background-color: white;}
}
```

Notice how the SystemTitle and ProcTitle class selectors are set to purple and red, respectively, but outside the @MEDIA rule sections. Inside the @MEDIA rule sections, you can see that the header and footer areas are made to have a background color of purple and a foreground color of white for the SCREEN rule. For the PRINT rule, the headers have a background of light gray (#aaaaaa) and a foreground color of black. The data cells also exhibit differences. For example, the data cells for PRINT rules are 9-point and Courier font, while the data cells for SCREEN rules are set to a relative size of "small" and to the sans-serif font of Arial. The background of the SCREEN data cell is a pale pink (#ffccff) while the background color for the PRINT data cell is white.

If you did not specify a particular @MEDIA rule for the CSSSTYLE= option, then ODS would use the SCREEN rule, if one was found. Otherwise, it would follow the rules of the CSS "cascade" to set the style for the class selectors in the output. Specifying the @MEDIA rule is simple; you just put the name of the @MEDIA rule in sub-option parentheses after the CSSSTYLE= option value, as shown below:

```
ods html path='c:\temp' (url=none)
    file="demo4_pr.html"
    cssstyle='demo4.css'(print);

ods rtf file="c:\temp\demo4_pr.rtf"
    cssstyle='c:\temp\demo4.css'(print)
    startpage=no;

ods pdf file="c:\temp\demo4_pr.pdf"
    cssstyle='c:\temp\demo4.css'(print)
    startpage=no;

proc print data=sashelp.class(obs=3);
  title 'Use @MEDIA Rules for PRINT';
run;
```

```
ods select variables;
proc contents data=sashelp.class;
  title 'List of Variables';
run;

ods _all_ close;

ods html path='c:\temp\' (url=none)
    file="demo4_scr.html"
    cssstyle='demo4.css'(screen);

ods rtf file="c:\temp\demo4_scr.rtf"
    cssstyle='c:\temp\demo4.css'(screen)
    startpage=no;

ods pdf file="c:\temp\demo4_scr.pdf"
    cssstyle='c:\temp\demo4.css'(screen)
    startpage=no;

proc print data=sashelp.class(obs=3);
  title 'Use @MEDIA Rules for SCREEN';
run;

ods select variables;
proc contents data=sashelp.class;
  title 'List of Variables';
run;

ods _all_ close;
```

If you examine the two different sets of SAS code, you can see that the only difference between the two invocations is the @MEDIA rule that is specified in parentheses as a sub-option to the CSSSTYLE option. For RTF and PDF, the STARTPAGE= option was set to NO, so that the effect of the SAS title and the procedure title style could be seen on one page. Figure 5 shows the results of the @PRINT rule being used for all three destinations and Figure 6 shows the results of the @SCREEN rule being used for all three destinations. You might have to experiment with different CSS style properties to find the combination of properties that looks the way you want for all three destinations. For example, the PADDING style property works for HTML, but MARGIN-RIGHT and MARGIN-LEFT style properties work better for RTF and PDF instead of PADDING. Again, your company Webmaster might have good CSS resources to help you design a CSS file with @MEDIA rules.

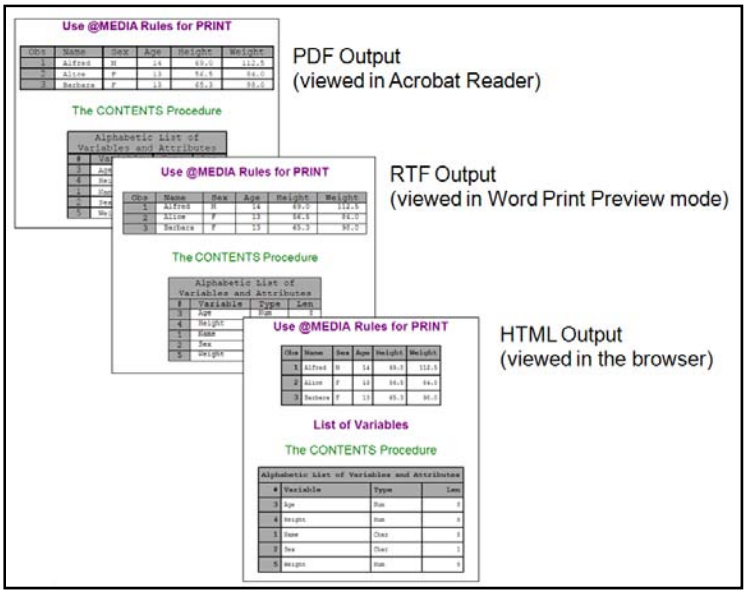

Figure 5: Using the PRINT @MEDIA CSS Rule

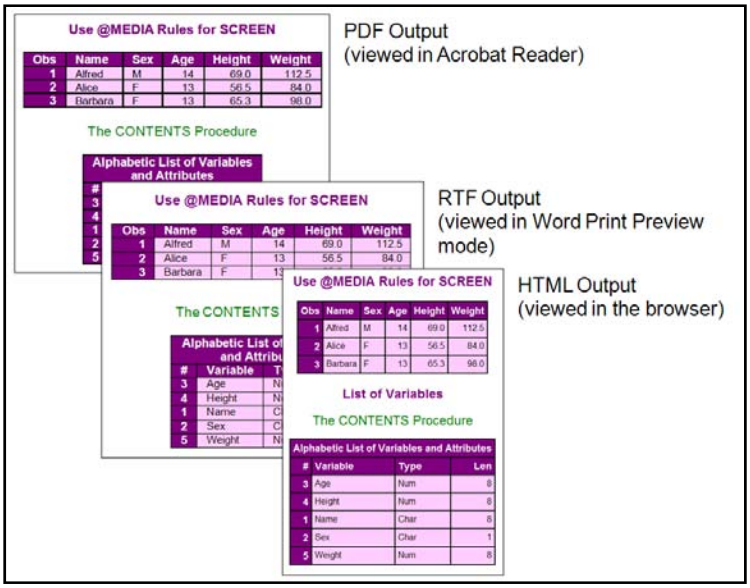

Figure 6: Using the SCREEN @MEDIA CSS Rule

When you run the demonstration code, you will be able to see better that the PRINT output from the @MEDIA rule uses the Courier New font, while the SCREEN output from the @MEDIA rule uses the Arial font. The SAS title and the procedure title for PROC CONTENTS are the same color and font in both all the outputs because they were specified once and not overridden in either of the @MEDIA rule sections.

CSS @IMPORT TECHNIQUE

Another way that CSS files are sometimes used is that multiple CSS files can be specified in multiple <LINK> tags in an HTML file. The CSS 2 specification defined the @IMPORT capability that allows one CSS file to contain pointers, or @IMPORT declarations, which allowed multiple CSS files to be stored in different locations, or even on different Web servers, and then the @IMPORT technique would cause the multiples CSS files to be brought together.

In the final example, the DEMO5.CSS file will contain an @MEDIA rule for the TTY device. Although your imported CSS could be as complicated as you want, this CSS File sets the data cells to have a background of CYAN and the header cells to have a gray background. Now, we need the DEMO5 CSS file, but we also need a CSS file with the appropriate @IMPORT declaration, as shown in the two code examples below:

DEMO5.CSS

```
@media tty{
.table
  {background-color: #eeeeee;
   border-spacing: 0;
   padding: 7px;
   border: 1px black solid;}
.proctitle
  {font-family: Courier New, Courier, serif;
   font-size: large;
   font-weight: 12pt;
   color:black;}
.systemtitle
  {font-family: Courier New, Courier, serif;
   font-size: large;
   font-weight: 14pt;
   color: black;}
.header, .rowheader, .footer, .rowfooter
  {color: black;
   font-size: 11pt;
   font-family: Courier New, Courier, serif;
   background-color: #dddddd;}
.data
  {font-size: 9pt;
   font-family: Courier New, Courier, serif;
   background-color: cyan;}
}
```

DEMO5_IMP.CSS

```
<style type="text/css">
<!--
@import "demo4.css";
@import "demo5.css";
-->
</style>
```

When invoked with our SAS code, we reference the DEMO5_IMP.CSS file, which imports both DEMO4.CSS (from the last example and DEMO5.CSS from this example. SAS and ODS will use both CSS files, resolve the style cascade in both files, and the resulting output (shown here in HTML) has the desired style characteristics. (The RTF and PDF files look essentially the same).

```
ods html path='c:\temp\' (url=none)
    file="demo5_two.html"
    cssstyle='demo5_imp.css'(tty);

ods rtf file="c:\temp\demo5_two.rtf"
    cssstyle='c:\temp\demo5_imp.css'(tty);

ods pdf file="c:\temp\demo5_two.pdf"
    cssstyle='c:\temp\demo5_imp.css'(tty);

proc print data=sashelp.class(obs=3);
  title 'Use Two CSS Files';
run;

ods _all_ close;
```

The HTML output from the above code is shown in Figure 7.

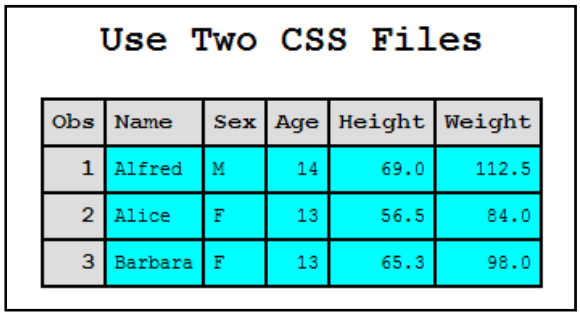

Figure 7: Using the @IMPORT CSS Declaration

CONCLUSION

If you are going to use the CSSSTYLE= option, you will benefit from further study of CSS syntax and capabilities. You will find that you have to test your CSS file in all the destinations where it will be used to make sure that all the style properties work the way you want them to. Jonathan Swift said "The proper words in the proper places are the true definition of style." Mr. Swift was about three hundred years too early to be using SAS, or I'm sure he would have added that the proper data and analysis in the proper places are the true definition of style. All the rest is cosmetics. Focus first on getting the right data and the right analysis, then focus on the cosmetics and the style for the output.

It's true with fashion that styles are always evolving and the ODS use of style is evolving, too. Using the CSSSTYLE= option, when coupled with the @MEDIA rules gives you a powerful new way to make an impact on your SAS output.

RECOMMENDED READING

Gebhart. Eric. 2007. "ODS Markup, Tagsets, and Styles! Taming ODS Styles and Tagsets." *Proceedings of the SAS Global Forum 2007 Conference*. Cary, NC. SAS Institute Inc. Available at www2.sas.com/proceedings/forum2007/225-2007.pdf.

Parker, Chevell. "Generating Custom Excel Spreadsheets using ODS." *Proceedings of the SAS Users Group International 28 Conference.* Cary, NC. SAS Institute Inc.
Available at www2.sas.com/proceedings/sugi28/012-28.pdf.

Smith, Kevin D. "The TEMPLATE Procedure Styles: Evolution and Revolution." *Proceedings of the SAS Users Group International 31 Conference.* Cary, NC. SAS Institute Inc. Available at www2.sas.com/proceedings/sugi31/053-31.pdf.

ACKNOWLEDGMENTS

The author thanks Chevell Parker, Bari Lawhorn, David Kelley, and Kevin Smith for graciously answering all ODS and CSSSTYLE questions. Thanks are also due to Amy Wolfe and Anne Harvey, whose editing skills have made this a much better paper.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author:

Cynthia L. Zender
SAS Institute Inc.
Work Phone: 575-522-3803
E-mail: Cynthia.Zender@sas.com