SAS® Style Templates: Always in Fashion

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

ABSTRACT

The syntax for style templates changed in SAS® 9.2 and became much easier to use. With the new CLASS statement, there is no need to debate the use of the STYLE statement versus the now defunct REPLACE statement. Yes, the REPLACE statement has gone away, and now we have the CLASS statement. In addition, you can import a CSS file into a SAS style template. All of these exciting new changes add up to stylish and fashionable output using ODS style templates.

This paper provides an introduction into the use of style templates in SAS 9.2. Methods of determining the correct style elements will be shown along with several concrete examples of making style template changes. The use of the IMPORT statement will also be demonstrated. In addition to these topics, a job aid will be provided that outlines the most commonly used style elements and their attributes.

INTRODUCTION

The first fact that most SAS users learn about the Output Delivery System (ODS) is the use of the basic "sandwich" invocation statements. Second, most SAS users learn about the use of the STYLE= option. The STYLE= option specifies an ODS style template (or style definition) which, for destinations that support styles, changes the overall look and feel of the output. Each ODS destination is like a major fashion designer or fashion house (Yves St. Laurent, Chanel, Calvin Klein, Armani), and each style template or style definition is like a particular outfit in a fashion line—the *prêt-a-porter* or ready to wear—outfits in the ODS fashion line styles are provided by SAS at installation time.

ODS HTML has a certain set of style conventions, as seen in STYLES.DEFAULT (sans-serif fonts, relative font sizes, more use of color). ODS PDF and ODS RTF have a different set of style conventions (serif fonts, absolute font sizes in points, gray and white colors) as seen in STYLES.PRINTER and STYLES.RTF, respectively.

When you use the TEMPLATE procedure to design your own style template for use with ODS, you move into the world of custom style templates. Most of the time, you can create your new style template by starting with an existing template as the basis for your new style. If you need something special, like designing a style template from scratch or using a Cascading Style Sheet (CSS) file as the starting point for your new style template, those are also possibilities.

In the fashion industry, custom-fitted clothing is known as *haute couture*. *Haute couture* translates to high sewing or high dressmaking, and when talking about clothing, it means clothes that are made to order for a specific customer or for the runway. These clothes are usually made with pricey fabric and are sewn with exquisite handsewn seams and finishes. When talking about style templates, the attention to detail that you find in *haute couture* translates to the attention that you must invest in learning and mastering TEMPLATE procedure syntax, as well as style element and style attribute concepts.

HOW STYLE TEMPLATES WORK

The style template is used by ODS to provide style information to the applications that render ODS output for viewing. A style template (or style definition) is a collection of style elements. These style elements can impact tabular and/or graphic portions of your output. (If you use the GSTYLE system option in SAS 9.2, there are graph style elements that will impact the graphic portions of your output. If you use the NOGSTYLE system option, then SAS/GRAPH® procedures such as GCHART, GPLOT, and GMAP will <u>not</u> use style template information. However, the new SG procedures, SGPLOT, SGSCATTER, and SGPANEL always use style template information.)

Some style elements are abstract elements—such as **Container** or **Cell**—whose purpose is to act as a parent element from which other style elements can inherit their style attributes. Non-abstract elements (such as **Header** or **SystemTitle**) represent style elements that directly correspond to a piece of your output. For example, the **Header** style element impacts the style of column headings in tabular output, and the **GraphLabelText** style element impacts the style of labels in graphical output. Each style element is a collection of style attributes. The style attributes represent common style characteristics (foreground color, background color, font face, font weight, font style) that should be used for a particular piece of output, as indicated by the style element usage or by style element inheritance.

Figure 1 shows an example of some tabular and graphic style elements and some of the style attributes that are defined by each element. For example, both the **Header** and the **GraphLabelText** elements have a font attribute and a color attribute. This means that it is possible for each style element to have separate font information (font face, font style, font weight, font size) or separate color information. The two color definitions are shown in Figure 1, note that the style attribute color is the same as the style attribute foreground in earlier versions of SAS.

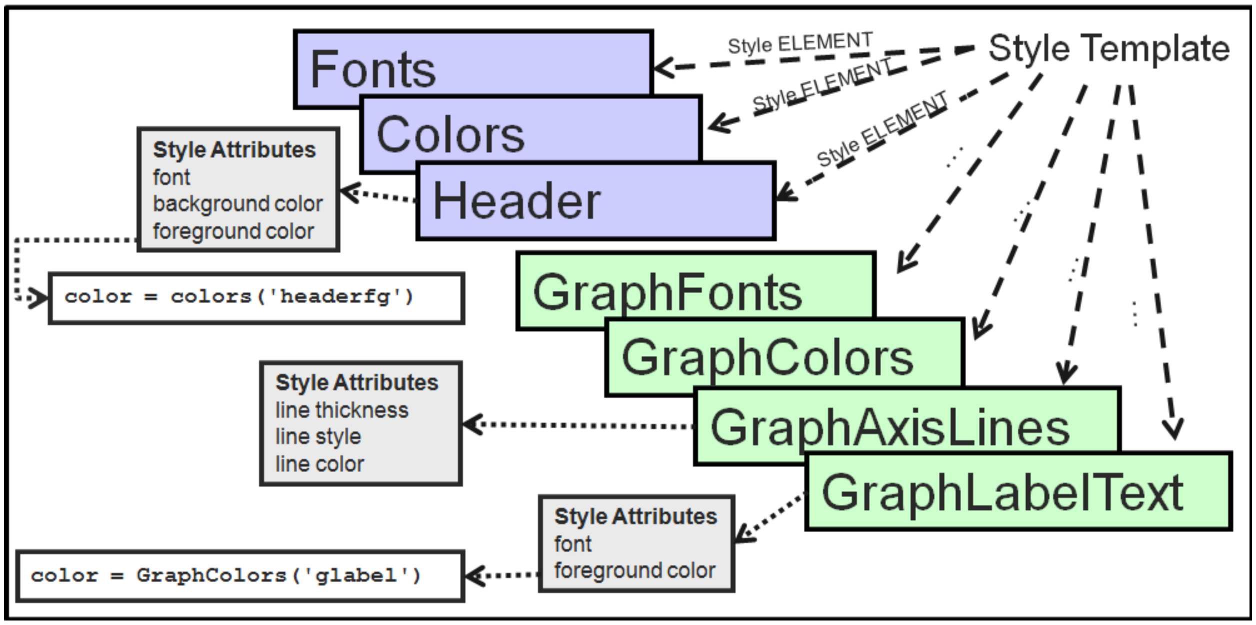


Figure 1. Conceptual relationship between style elements and style attributes

If we investigated further in STYLES.DEFAULT, we would discover that the color and font information for these two style elements is:

Header: `color=colors('headerfg')`
```
        font = fonts('HeadingFont')
```

GraphLabelText: `color=GraphColors('glabel')`
```
                font = GraphFonts('GraphLabelFont')
```

The syntax `color=colors('...')` for the color attribute is a way for the style template to point to a specific, named color in the **colors** style element or the **GraphColors** style element, as shown below (from STYLES.DEFAULT). Compare the **GraphColors** style element with the **colors** style element, and you will see specific color names provided in the **GraphColors** style element, while the **colors** style element contains references to the **color_list** style element. Only the style attributes for **Header** and **GraphLabelText** are highlighted.

One useful side effect of putting colors in a lookup list in the style template is that if you want to have an impact on all the colors in the style template, you only need to change the values in the **color_list** style element. If you change the colors here, you would effectively change every style element that used `fgA2`, for example.

On the other hand, if you only wanted to change the **Header** style element and any elements that inherited from the **Header** style element, then you could leave the **color_list** element unchanged, and make your changes directly in the CLASS statement for the **Header** element.

GraphColors style element	colors and color_list style elements
`class GraphColors /`	`class color_list /`
`    . . . more style attributes . . .`	`    'fgB2' = cx0066AA`
`    'gcdata' = cx000000`	`    'fgB1' = cx004488`
`    'gdata' = cxB9CFE7`	`    'fgA4' = cxAAFFAA`
`    . . . more style attributes . . .`	`    'bgA4' = cx880000`
`    'gtext' = cx000000`	`    'bgA3' = cxD3D3D3`
`    'glabel' = cx000000`	`    'fgA2' = cx0033AA`
`    'gborderlines' = cx000000`	`    'bgA2' = cxB0B0B0`
`    'goutlines' = cx000000`	`    'fgA1' = cx000000`
`    'ggrid' = cxECECEC`	`    'bgA1' = cxF0F0F0`
`    . . . more style attributes . . .;`	`    'fgA' = cx002288`
	`    'bgA' = cxE0E0E0;`
	`class colors /`
	`    . . . more style attributes . . .`
	`    'headerfg' = color_list('fgA2')`
	`    'headerbg' = color_list('bgA2')`
	`    . . . more style attributes . . .`
	`    'systitlefg' = color_list('fgA')`
	`    'systitlebg' = color_list('bgA')`
	`    . . . more style attributes . . .`
	`    'docfg' = color_list('fgA')`
	`    'docbg' = color_list('bgA');`

When you look at the **GraphColors** style element, you can see that the setting for the `glabel` attribute, cx000000, is what will be used for the GraphLabelText style element. However, when you look at the **colors** style element, you see that the **Header** foreground color is cx0033AA. (To find out the color for the **Header** style element, you have to trace from the `headerfg` value to the `fgA2` attribute value). This type of inheritance (using a lookup list) only happens in style templates with colors and fonts.

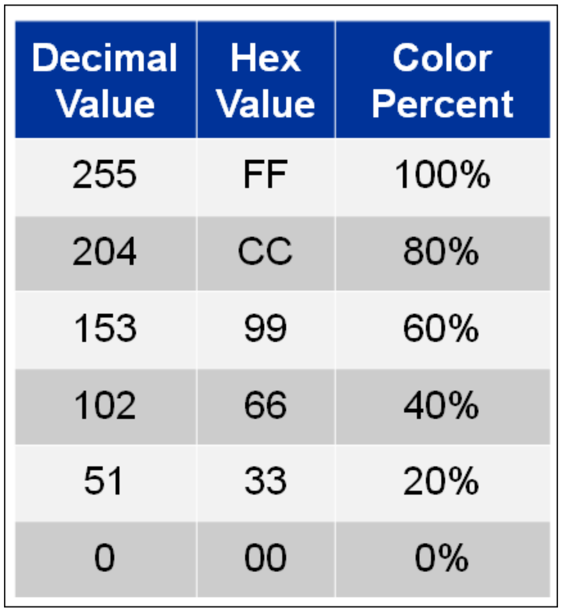
There are two other types of inheritance that you could encounter in a style template. One type is style element inheritance, such as when the **Header** element inherits from the **HeadersandFooters** style element, or the **RowHeader** style element inherits from the **Header** style element. The other type of inheritance is style template inheritance, such as when STYLES.PRINTER inherits from STYLES.DEFAULT or STYLES.RTF inherits from STYLES.PRINTER.

Starting with SAS 9.2, you don't have to worry too much about any of these types of style template, style element and style attribute inheritance, because the developers have devised a clever way to make sure that the inheritance structure is maintained. They've created a style template called BASE.TEMPLATE.STYLE, which is the implicit parent template for all of the style templates that you create. This is one of the reasons that you don't have to worry about inheritance very much; because when you create a style template, the pattern for style element inheritance has already been designed.

For more information about other ODS templates, such as Table, Tagset, and Graph templates, refer to my SAS Global Forum 2009 paper entitled, "Tiptoe Through the Templates," available from this URL: http://support.sas.com/resources/papers/proceedings09/227-2009.pdf or from the http://support.sas.com/rnd/papers Web site.

UNDERSTANDING COLORS

Let's talk about colors for a minute. You might wonder about the colors that are going to be used for the **Header** style element or the **GraphLabelText** style element. The colors are specified as RGB (Red-Green-Blue) color values. In the RGB color system, color names are of the form CXrrggbb, where the color values are given as hexadecimal numbers in the range 00 through FF. Most web page colors and output for the web use RGB color values. SAS style templates use RGB color codes. The color specification can be deconstructed as follows: CX indicates the RGB color specification; rr is the red component; gg is the green component; bb is the blue component. Because the digits or characters are hexadecimal values, you can think of them as percentages, where the values represent these percentages. Figure 2 shows some examples of hexadecimal values, the equivalent decimal numbers, and their color percentages.



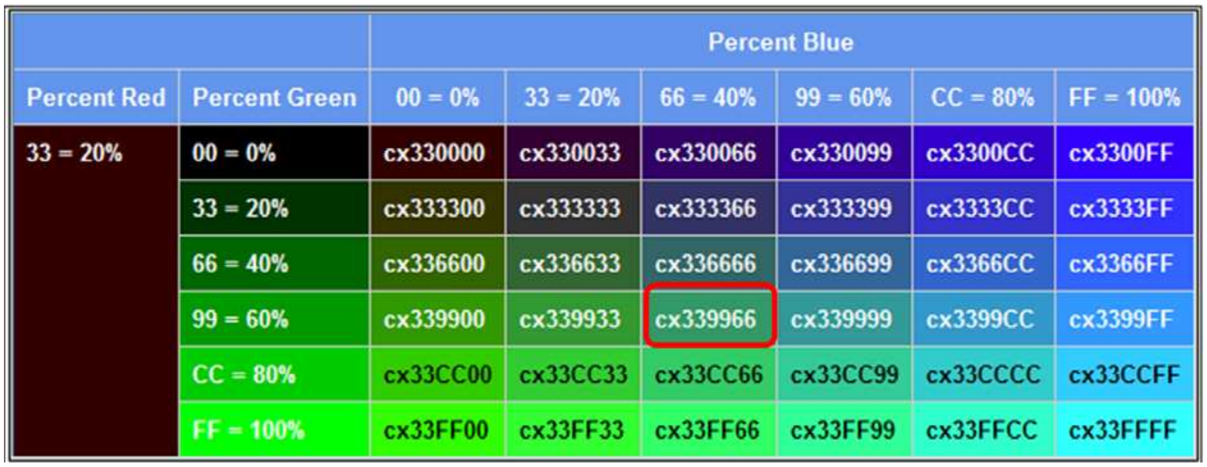
Decimal Value	Hex Value	Color Percent
255	FF	100%
204	CC	80%
153	99	60%
102	66	40%
51	33	20%
0	00	0%

Figure 2. Hex values for colors with decimal and percent equivalents

Using this information, you now have a way to translate the colors in a general fashion.

cxFF0000 = 100% red, 0% green, 0% blue
cx00FF00 = 0% red, 100% green, 0% blue
cx339966=20% red, 60% green, 40% blue

Colors that are easy to remember are CX000000 (black – the absence of all colors) and CXFFFFFF (white – the presence of all colors). Otherwise, to figure out what an actual hexadecimal RGB color looks like (such as CX339966), you either have to consult a color map or an online color application, or you need to experiment with color values until you find a color that suits your purpose. In the zip file of programs that accompanies this paper (see "Download Information" at the end of this paper), you will find a program that creates a color map. This color map is partially shown in Figure 3, where CX339966 is revealed to be a sort of teal-blue-green color.



		Percent Blue					
Percent Red	Percent Green	00 = 0%	33 = 20%	66 = 40%	99 = 60%	CC = 80%	FF = 100%
33 = 20%	00 = 0%	cx330000	cx330033	cx330066	cx330099	cx3300CC	cx3300FF
	33 = 20%	cx333300	cx333333	cx333366	cx333399	cx3333CC	cx3333FF
	66 = 40%	cx336600	cx336633	cx336666	cx336699	cx3366CC	cx3366FF
	99 = 60%	cx339900	cx339933	cx339966	cx339999	cx3399CC	cx3399FF
	CC = 80%	cx33CC00	cx33CC33	cx33CC66	cx33CC99	cx33CCCC	cx33CCFF
	FF = 100%	cx33FF00	cx33FF33	cx33FF66	cx33FF99	cx33FFCC	cx33FFFF

Figure 3. Partial program output of color values and colors

UNDERSTANDING FONTS

Fonts are referenced in style templates using the same lookup list technique that was used for color references. The font reference is straightforward once you understand that the style attribute for fonts usually refers to SAS registry font specifications. For example, the **Header** font specification in STYLES.DEFAULT refers to the HeadingFont attribute, which is `'headingFont' = ("<sans-serif>, Helvetica, sans-serif",4,bold).` This means that the first choice for the **Header** font will be the sans-serif font defined in the SAS registry (on my Windows system, this is the Arial font). Then, if Arial is not available when the output is rendered, Helvetica will be used. If Helvetica is not available on the rendering system, any sans-serif font on the system will be used. If all else fails, the application will use its default font for rendering. The size for the **Header** font is going to be a relative size of 4 (HTML relative fonts are specified in sizes from 1 to 7), and the font weight will be bold. In the style templates provided by SAS, HTML-based fonts are specified with relative numbers, and the fonts used for the Printer and RTF destinations are specified with absolute point sizes.

You can use the FONTREG procedure to register fonts for use with SAS/GRAPH (and all other SAS products). However, a discussion of the FONTREG procedure is outside the scope of this paper. If you do register True Type fonts (not listed in the SAS registry), you only need to reference the correct font name in order for that font to be used. Remember, however, that in your templates, it is always a good idea to specify a secondary font or a font family as an alternative in case a report is opened or rendered on a system without the first font you specify.

STYLE TEMPLATE FASHION SHOW

In SAS 9.2, you can use the CLASS or STYLE statement to change the attributes of an existing style element such as **Header**, and/or the IMPORT statement to convert a CSS file to a style template. To illustrate the use of the CLASS, STYLE, and IMPORT statements (among other things), suppose that you're sitting in the audience at the House of ODS fashion show, where I have my *haute couture* custom collection of templates strutting down the fashion runway for your review.

You might not want to buy all the outfits, but perhaps some of these style template *couture* techniques will be useful to you. The fashions on display today will illustrate changes that use the CLASS statement and automatic inheritance, changes that bypass inheritance, changes to simple style elements (like **Header**, **RowHeader** or **SystemTitle**), and changes to more complex elements such as the **Table** or **Output** elements. Along the way, you'll learn about some of the custom detailing and style attributes used in the creation of these templates.

On the runway today are:

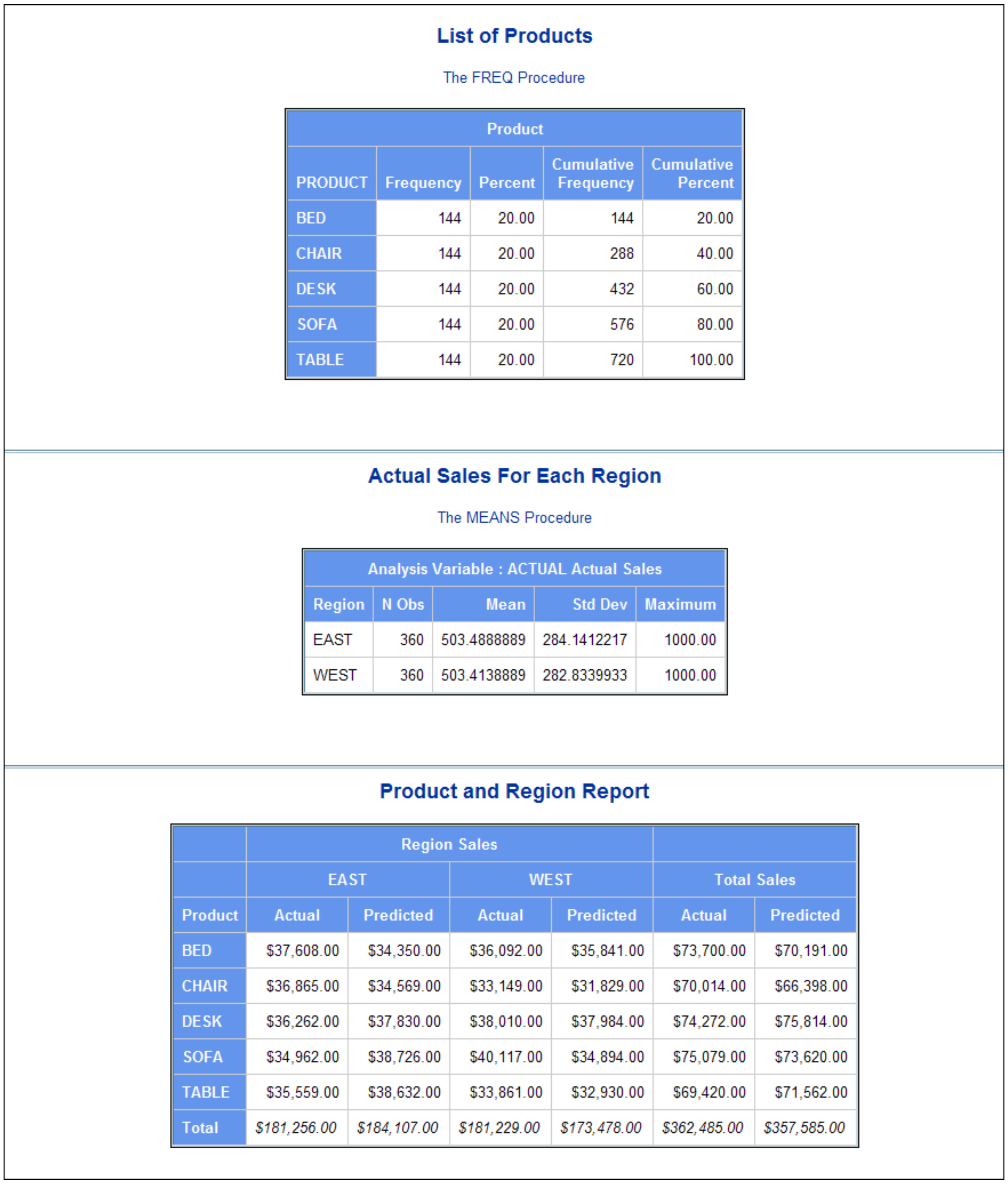
Outfit 1: Changing the **Header** and **RowHeader** style
Outfit 2: Changing **Header** style, but using default colors for **RowHeader**
Outfit 3: Making TITLE statements different sizes
Outfit 4: Changing the default formatting for output from the ODS TEXT= option
Outfit 5: Changing interior table lines
Outfit 6: Using CSS class selectors to change **Header** style
Outfit 7: Changing graph output with a style template

UNDERSTANDING THE PATTERN AND DIRECTIONS

Whenever you make a high fashion garment (or any garment for that matter), one technique is to start with a muslin shell in the basic pattern that you want, and then modify the muslin pattern to contain the changes that you are designing. For this fashion show, the basic muslin that is the following example uses STYLES.SASWEB as the basic style template. For most of the outfits, this is our basic starting point. The starting output is shown in Figure 4. The style STYLES.SASWEB is designed for use with HTML-based destinations. Most of the style templates will be used with the ODS HTML destination. However, the methods of modifying the style syntax and the information about style elements and attributes are applicable to other destinations as well. Several of the template designs will be shown with the ODS PDF or ODS RTF destination.

For more information about style template syntax changes between SAS 9.1.3 and SAS 9.2, refer to Kevin Smith's SAS Global Forum paper (Paper 053-31) entitled, "The TEMPLATE Procedure Styles: Evolution and Revolution." (See “References” at the end of this paper.) An invaluable reference is the ODS documentation on the TEMPLATE procedure, because no paper can cover the wealth of information and details that are contained in the ODS documentation. The best advice I ever got when first learning SAS and JCL (at the same time!) was "RTFM.". Sometimes the acronym was an expletive, and sometimes an encouragement to read the manual in question. Some documentation Web sites that I constantly refer to are listed below.

- For information about using style attributes and style elements together: http://support.sas.com/documentation/cdl/en/odsug/61723/HTML/default/a002685135.htm.
- For information about style attributes and their values: http://support.sas.com/documentation/cdl/en/odsug/61723/HTML/default/a002972093.htm.
- For documentation on ODS Style Elements for Device-based Graphics and Template-based Graphics:
  Device-based graphics: http://support.sas.com/documentation/cdl/en/odsug/61723/HTML/default/a003241787.htm
  Template-based graphics: http://support.sas.com/documentation/cdl/en/odsug/61723/HTML/default/a003166123.htm.



List of Products

The FREQ Procedure

Product				
PRODUCT	Frequency	Percent	Cumulative Frequency	Cumulative Percent
BED	144	20.00	144	20.00
CHAIR	144	20.00	288	40.00
DESK	144	20.00	432	60.00
SOFA	144	20.00	576	80.00
TABLE	144	20.00	720	100.00

Actual Sales For Each Region

The MEANS Procedure

Analysis Variable : ACTUAL Actual Sales				
Region	N Obs	Mean	Std Dev	Maximum
EAST	360	503.4888889	284.1412217	1000.00
WEST	360	503.4138889	282.8339933	1000.00

Product and Region Report

	Region Sales					
	EAST		WEST		Total Sales	
Product	Actual	Predicted	Actual	Predicted	Actual	Predicted
BED	$37,608.00	$34,350.00	$36,092.00	$35,841.00	$73,700.00	$70,191.00
CHAIR	$36,865.00	$34,569.00	$33,149.00	$31,829.00	$70,014.00	$66,398.00
DESK	$36,262.00	$37,830.00	$38,010.00	$37,984.00	$74,272.00	$75,814.00
SOFA	$34,962.00	$38,726.00	$40,117.00	$34,894.00	$75,079.00	$73,620.00
TABLE	$35,559.00	$38,632.00	$33,861.00	$32,930.00	$69,420.00	$71,562.00
Total	*$181,256.00*	*$184,107.00*	*$181,229.00*	*$173,478.00*	*$362,485.00*	*$357,585.00*

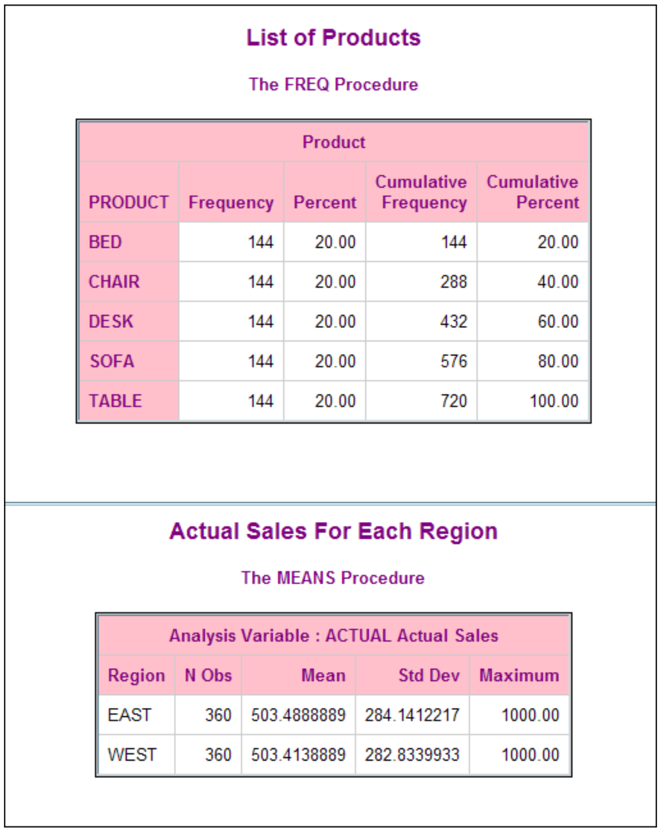
Figure 4. Report Starting Point

The program is very basic code for the FREQ, MEANS, and REPORT procedures. The data used is a subset of SASHELP.PRDSALE, and the ODS invocation is:

```
ods html file='demo00_basic.html' style=sasweb;
. . . proc freq, proc means and proc report steps . . .
ods html close;
```

OUTFIT 1: CHANGING HEADER AND ROWHEADER STYLE

For the first example, only the FREQ and MEANS procedures are used. This example illustrates how to make changes to the **Header** style element and the **RowHeader** style element. In addition, the template code also changes the **ProcTitle** and **SystemTitle** style elements.



List of Products

The FREQ Procedure

Product				
PRODUCT	Frequency	Percent	Cumulative Frequency	Cumulative Percent
BED	144	20.00	144	20.00
CHAIR	144	20.00	288	40.00
DESK	144	20.00	432	60.00
SOFA	144	20.00	576	80.00
TABLE	144	20.00	720	100.00

Actual Sales For Each Region

The MEANS Procedure

Analysis Variable : ACTUAL Actual Sales				
Region	N Obs	Mean	Std Dev	Maximum
EAST	360	503.4888889	284.1412217	1000.00
WEST	360	503.4138889	282.8339933	1000.00

Figure 5. Program Results

The template code that produced the above results is shown below. Only the ODS HTML sandwich statements are shown for the PROC MEANS and PROC FREQ steps that use STYLES.DEMO01.

```
ods path work.temp(update)
         sashelp.tmplmst(read);

proc template;
  define style styles.demo01;
  parent=styles.sasweb;
  class Header/
    background=pink
    foreground=purple;
  class ProcTitle /
    foreground=purple
    font=fonts('HeadingFont');
  class SystemTitle /
    foreground=purple;
  end;
run;
```

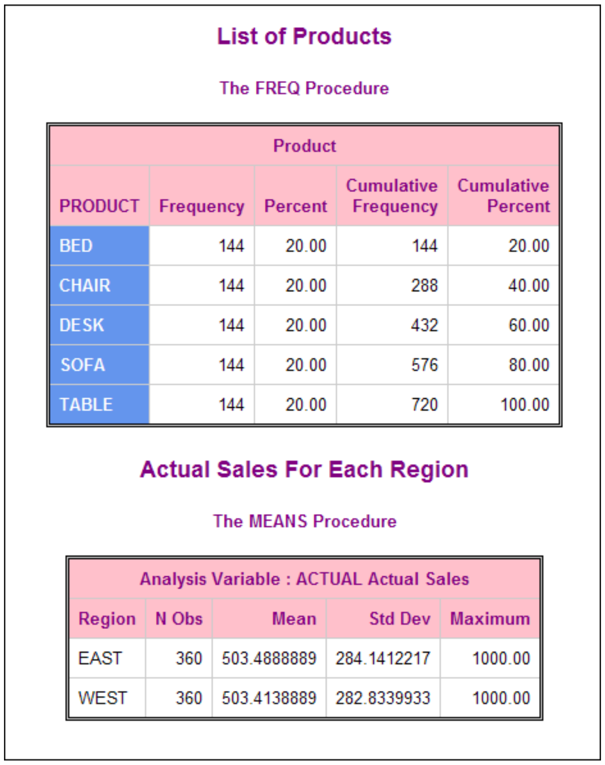
```
ods html file='demo01_outfit1.html' style=styles.demo01;
  . . . proc means and proc freq steps . . .
ods html close;
```

This is a very basic template. The STYLES.SASWEB style is used as the parent template. Note that you do not have to worry about the **color_list** or the **fonts** style elements. The CLASS statement works with the way that style element inheritance is resolved to ensure that the style attributes listed in the code override the same-named style attributes for the style element from the parent template. In earlier versions of SAS, you had to know whether to use a STYLE statement or a REPLACE statement in order to override style attributes. However, in SAS 9.2, the CLASS statement is all you need to use. (One of the ODS developers calls the CLASS statement "syntactic sugar for STYLE foo FROM foo".) The font characteristics of the procedure title are inherited from the HeadingFont attribute, while the foreground color comes directly from the foreground attribute override. (Note that you could have used `color=purple` as well, because foreground and color are both acceptable style attributes for changing foreground color, just as background and backgroundcolor are both acceptable style attributes for changing background color.)

OUTFIT 2: CHANGING HEADER STYLE, BUT USING DEFAULT COLORS FOR ROWHEADER

The header area of a SAS report, whether created with the MEANS, FREQ, or REPORT procedure, is the area that spans the report from left to right along the top of the report table. For the MEANS procedure, in our output, it is the area of the report that contains the class variable name and the calculated statistics. The header area for the FREQ procedure contains the variable name from the TABLE statement and the frequency and percent statistics. The FREQ procedure also has row header area. In our report, it is the leftmost column, which contains the names of each product. In the first example, the row header area changed when the header area changed because the **RowHeader** style element inherits from the **Header** style element. What if you wanted to change the header area of each table to the current color scheme but did not want to change the row header area?

There are actually several ways to achieve this result. Only a few of them are shown below. (The other methods are shown in the zip file of programs that you can download.) No matter which method you use, you can achieve the results shown in Figure 6. Because the MEANS procedure does not have a row header area, the new style template has no effect on the MEANS procedure output. But, with the FREQ procedure, you can see how the row header area uses the regular STYLES.SASWEB colors, while the header area uses the new colors.



List of Products

The FREQ Procedure

Product				
PRODUCT	Frequency	Percent	Cumulative Frequency	Cumulative Percent
BED	144	20.00	144	20.00
CHAIR	144	20.00	288	40.00
DESK	144	20.00	432	60.00
SOFA	144	20.00	576	80.00
TABLE	144	20.00	720	100.00

Actual Sales For Each Region

The MEANS Procedure

Analysis Variable : ACTUAL Actual Sales				
Region	N Obs	Mean	Std Dev	Maximum
EAST	360	503.4888889	284.1412217	1000.00
WEST	360	503.4138889	282.8339933	1000.00

Figure 6. Program Results

```
proc template;
  define style styles.demo02a;
  parent=styles.sasweb;
  class Header/
    background=pink
    foreground=purple;
  class RowHeader /
    background=color_list('bgA1')
    foreground=color_list('bgA');
  class ProcTitle /
    foreground=purple
    font=fonts('HeadingFont');
  class SystemTitle /
    foreground=purple;
  end;
run;

ods html file='demo02a_outfit2.html' style=styles.demo02a;

  . . . proc means and proc freq steps . . .

ods html close;
```

The changes to the **Header**, **SystemTitle**, and **ProcTitle** elements are the same as in the previous program. A CLASS statement has been added for the **RowHeader** element. To figure out the correct **color_list** attribute to use, you need to do a bit of investigation in STYLES.SASWEB. Because you already know (or can find out) that the **RowHeader** element inherits attributes from the **Header** element, you only need to do a bit of tracking to discover that the appropriate **color_list** attributes are `bgA` and `bgA1`. The relevant sections from STYLES.SASWEB are shown below.

```
        style color_list/
           . . . more attributes . . .
           'bgA1' = cx6495ED
           'bgA' = cxffffff;
        style colors /
           . . . more attributes . . .
           'headerfg' = color_list('bgA')
           'headerbg' = color_list('bgA1') ;
        style HeadersAndFooters from Cell
           "Abstract. Controls table headers and footers." /
           font = fonts('HeadingFont')
           color = colors('headerfg')
           backgroundcolor = colors('headerbg');
        style Header from HeadersAndFooters
           "Controls the headers of a table.";
        style RowHeader from Header;
```

An even simpler way to create a style template that bypasses inheritance for the row header area on the report would be to let inheritance work for you by creating the new style template with the syntax shown below. The results from this code would be the same as shown in Figure 6.

```
proc template;
  define style styles.demo02b;
  parent=styles.sasweb;
  class Header /
    background=pink
    foreground=purple;
  class RowHeader from HeadersandFooters/;
  class ProcTitle /
    foreground=purple
    font=fonts('HeadingFont');
  class SystemTitle /
```

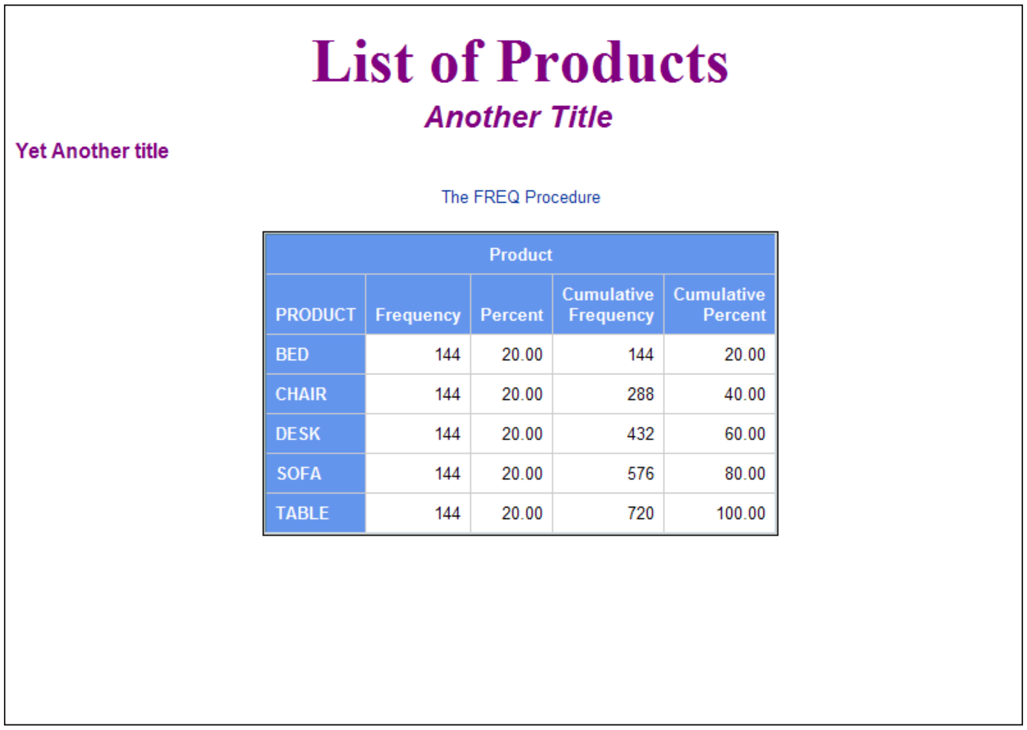
```
        foreground=purple;
      end;
    run;
```

In the above example, you would just set the **RowHeader** style element to explicitly inherit from the **HeadersandFooters** style element instead of the **Header** element. This would also allow the header area to be pink and purple, while leaving the row header area unchanged and set to the original colors.

OUTFIT 3: MAKING TITLE STATEMENTS DIFFERENT SIZES

By default, when you use ODS style templates, all of the titles and footnotes are the same size. Sometimes, however, you want the different TITLE statements to be different sizes. You already know about the **SystemTitle** element, but when you change it, you change all of the titles (or footnotes via **SystemFooter**) to be the same. If you look in BASE.TEMPLATE.STYLE (as shown in the excerpt below), you will discover that the developers have planned ahead. In that template definition, there are placeholder style elements for the TITLE and FOOTNOTE statements that you could issue. The relevant **SystemTitle** elements are shown below (parallel style elements exist for **SystemFooter** elements). You would never change the Base.Template.Style template, of course. But, if you create a new style template that uses some of these elements (as shown below in styles.demo03), you can achieve the results shown in Figure 7.

```
proc template;
   define style Base.Template.Style;
    notes "Implicit parent for all style templates";
      . . . more style elements . . .
     style SystemTitle from TitlesAndFooters
            "Controls system title text.";
    style SystemTitle2 from SystemTitle
            "Controls system title2 text";
    style SystemTitle3 from SystemTitle2
            "Controls system title3 text";
      . . . more style elements . . .
   end;
run;
```



Product				
PRODUCT	Frequency	Percent	Cumulative Frequency	Cumulative Percent
BED	144	20.00	144	20.00
CHAIR	144	20.00	288	40.00
DESK	144	20.00	432	60.00
SOFA	144	20.00	576	80.00
TABLE	144	20.00	720	100.00

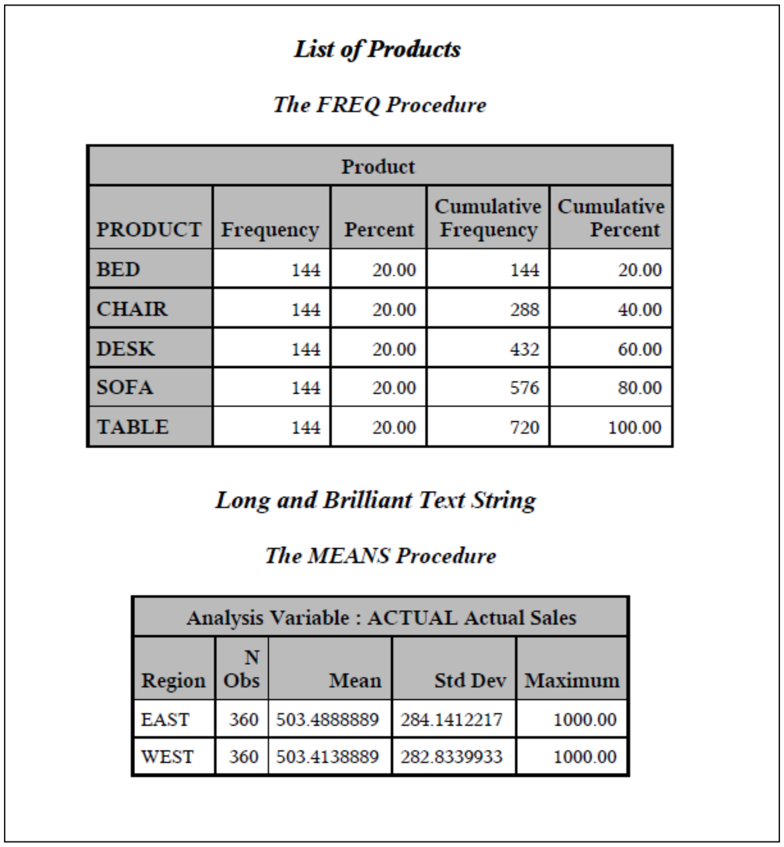
Figure 7. Program Results

```
proc template;
  define style styles.demo03;
  parent=styles.sasweb;
  class SystemTitle / foreground=purple
    font=("<serif>, Times New Roman, serif",7,bold);
  class SystemTitle2 / foreground=purple
    font=("<sans-serif>, Helvetica, sans-serif",5,bold italic);
  class SystemTitle3  /  foreground=purple just=l
    font=("<sans-serif>, Helvetica, sans-serif",3,bold);
  end;
run;
```

Note how the font and size are different for each of the TITLE statements in the output. Also, because of the `just=l` attribute change, the TITLE3 output is left-justified in the output. Because only the TITLE statements are of interest, only the FREQ procedure output is shown in Figure 7.

OUTFIT 4: CHANGING THE DEFAULT FORMATTING FOR OUTPUT FROM THE ODS TEXT= OPTION

You can use the ODS TEXT= statement to place text before, after, or between the output of procedure steps. The ODS TEXT= output is normally left-justified and in a regular-sized font. The style element that controls the ODS TEXT= text string is the **UserText** style element. Figure 8 shows how the ODS TEXT= string has been modified to act as a pseudo-title between the PROC FREQ and the PROC MEANS steps. Figure 8 shows the output from the ODS PDF destination, which is one of the destinations where the ODS TEXT= statement is frequently used to simulate a title between two procedure steps. The program for this example, however, contains the style template that you would use to format the **UserText** element for both HTML and RTF.



List of Products

The FREQ Procedure

Product				
PRODUCT	Frequency	Percent	Cumulative Frequency	Cumulative Percent
BED	144	20.00	144	20.00
CHAIR	144	20.00	288	40.00
DESK	144	20.00	432	60.00
SOFA	144	20.00	576	80.00
TABLE	144	20.00	720	100.00

Long and Brilliant Text String

The MEANS Procedure

Analysis Variable : ACTUAL Actual Sales				
Region	N Obs	Mean	Std Dev	Maximum
EAST	360	503.4888889	284.1412217	1000.00
WEST	360	503.4138889	282.8339933	1000.00

Figure 8. Program Results

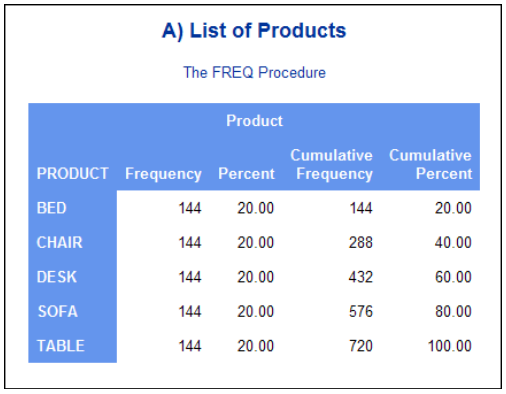
```
proc template;
  define style styles.demo04;
  parent=styles.printer;
  class UserText from SystemTitle/ just=c;
  end;
run;

ods pdf file='demo04_outfit4.pdf' style=styles.demo04 startpage=no;
  . . . PROC FREQ step . . .
ods text = '   ';
ods text = 'Long and Brilliant Text String';
  . . . PROC MEANS step . . .
ods _all_ close;
```

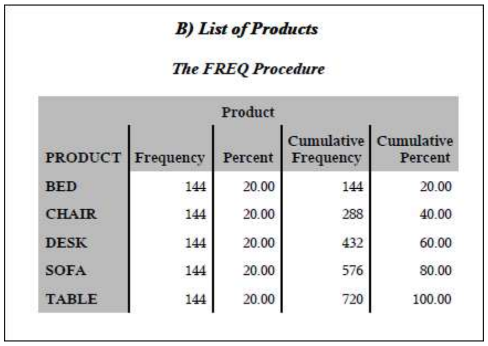
For the ODS PDF destination, the style template inherits from the STYLES.PRINTER style template. (You can also use this template for ODS RTF output.), If you want to use the ODS TEXT= text as a pseudo-title, then you only need to have the **UserText** element inherit attributes from the **SystemTitle** style element. Although you could specify all of the individual style attributes for **UserText**, the only specific attribute you need to specify is the just attribute because without this attribute, your ODS TEXT= string would still be left-justified.

OUTFIT 5: CHANGING INTERIOR TABLE LINES

There are several style attributes that work together to change the interior table lines in SAS output. By default, the interior table lines are controlled by the **Table** style element. The two essential attributes to change are the rules attribute and the borderspacing attribute (also known as the cellspacing attribute). In these examples, the table frame is turned off (`frame=void`) so that the effect of the rules attribute is easier to see. In Figure 9, the changed style template is shown next to the FREQ procedure output that uses the changed template.

Template Code	PROC FREQ Output

`proc template;`
`  define style styles.demo05a;`
`  parent=styles.sasweb;`
`  class table /`
`        frame=void`
`        borderspacing=0`
`        rules = none;`
`  end;`
`run;`
	

`proc template;`
`  define style styles.demo05b;`
`  parent=styles.printer;`
`  class table /`
`        frame=void`
`        borderspacing=0`
`        rules = cols;`
`  end;`
`run;`
	

A) List of Products

The FREQ Procedure

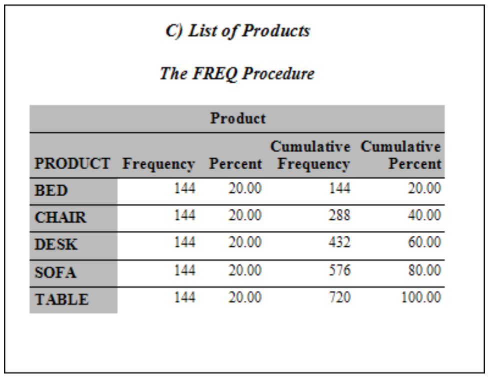
Product				
PRODUCT	Frequency	Percent	Cumulative Frequency	Cumulative Percent
BED	144	20.00	144	20.00
CHAIR	144	20.00	288	40.00
DESK	144	20.00	432	60.00
SOFA	144	20.00	576	80.00
TABLE	144	20.00	720	100.00

B) List of Products

The FREQ Procedure

Product				
PRODUCT	Frequency	Percent	Cumulative Frequency	Cumulative Percent
BED	144	20.00	144	20.00
CHAIR	144	20.00	288	40.00
DESK	144	20.00	432	60.00
SOFA	144	20.00	576	80.00
TABLE	144	20.00	720	100.00

```
proc template;
  define style styles.demo05c;
  parent=styles.rtf;
  class table /
        frame=void
        borderspacing=0
        rules = rows;
  end;
run;
```



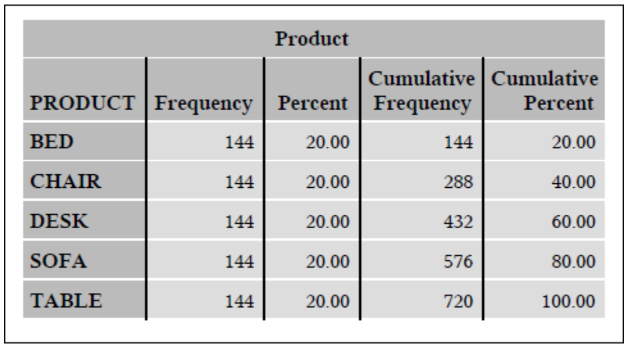
C) List of Products

The FREQ Procedure

Product				
PRODUCT	Frequency	Percent	Cumulative Frequency	Cumulative Percent
BED	144	20.00	144	20.00
CHAIR	144	20.00	288	40.00
DESK	144	20.00	432	60.00
SOFA	144	20.00	576	80.00
TABLE	144	20.00	720	100.00

Figure 9. Program Code and Results

In these three templates, when the table frame is turned off, you can see the effect of the three different specifications for the rules style attribute. The borderspacing attribute is set to zero so that the background of the table does not show through the normal space between each cell. For example, consider the same template code, but with a value of `0.5pt` for the borderspacing attribute and a background color of light gray for the data cells. What appears as white horizontal lines is the `0.5pt` space between each cell.



Product				
PRODUCT	Frequency	Percent	Cumulative Frequency	Cumulative Percent
BED	144	20.00	144	20.00
CHAIR	144	20.00	288	40.00
DESK	144	20.00	432	60.00
SOFA	144	20.00	576	80.00
TABLE	144	20.00	720	100.00

Figure 10. Program Results

There are many possible combinations of the rules and frame attributes. Remember that the JOURNAL and JOURNAL2 style templates are already designed without interior table lines and produce output that might be suitable for print and academic journal publications. Figure 11 shows the possible values that you can specify for each of these attributes. You'll need to experiment with different values in each destination to see which attributes give you the interior table lines you desire.

Attribute	Sample Values
RULES	ALL, COLS, GROUPS,NONE, ROWS
FRAME	ABOVE, BELOW, BOX, HSIDES, LHS, RHS, VOID, VSIDES

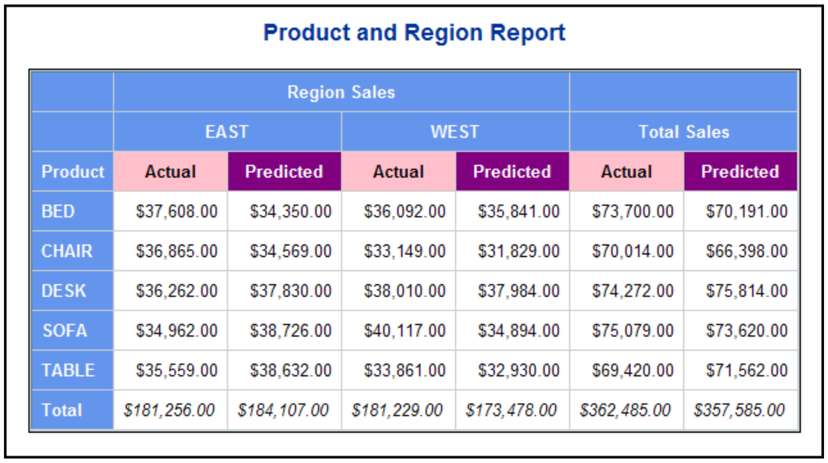
Figure 11. Attribute Values for RULES and FRAME Style Attributes

OUTFIT 6: USING CSS CLASS SELECTORS TO CHANGE HEADER STYLE

When you create ODS HTML destination output, the default behavior is for the style template attributes to be translated to CSS style properties and inserted into an in-line or embedded <STYLE> section in the HTML file. The RTF and PDF destinations do not work this way. Starting in SAS 9.2, however, you can use a CSS style sheet with the ODS RTF and ODS PDF destinations using the CSSSTYLE= option. Another way to use a CSS file is to import it into an ODS style template. Consider the REPORT procedure output shown in Figure 4. What if you wanted different styles for the Actual and Predicted variable headers? Consider the following CSS file (Hdr.CSS). It defines two class selectors, AHDR and PHDR.

```
.AHDR
{   font-family: Arial, Helvetica, sans-serif;
    font-size: x-small; font-weight: bold; font-style: normal;
    color: #000000; background-color: pink;}
.PHDR
{   font-family: Arial, Helvetica, sans-serif;
    font-size: x-small; font-weight: bold; font-style: normal;
    color: #000000; background-color: purple;}
```

The CSS file, when imported into a style template, adds two style elements to the new style template. These style elements will not be requested by any of the other ODS report areas (such as **SystemTitle**, **Header,** or **Data** elements). However, with the REPORT procedure, we can use the following syntax for the Actual and Predicted variables to produce the HTML results in Figure 12. The full text of the template and the REPORT procedure code follows Figure 12.



Product and Region Report

	Region Sales					
	EAST		WEST		Total Sales	
Product	Actual	Predicted	Actual	Predicted	Actual	Predicted
BED	$37,608.00	$34,350.00	$36,092.00	$35,841.00	$73,700.00	$70,191.00
CHAIR	$36,865.00	$34,569.00	$33,149.00	$31,829.00	$70,014.00	$66,398.00
DESK	$36,262.00	$37,830.00	$38,010.00	$37,984.00	$74,272.00	$75,814.00
SOFA	$34,962.00	$38,726.00	$40,117.00	$34,894.00	$75,079.00	$73,620.00
TABLE	$35,559.00	$38,632.00	$33,861.00	$32,930.00	$69,420.00	$71,562.00
Total	*$181,256.00*	*$184,107.00*	*$181,229.00*	*$173,478.00*	*$362,485.00*	*$357,585.00*

Figure 12. Program Results

```
ods path work.temp(update)
         sashelp.tmplmst(read);

proc template;
  define style styles.demo06;
    parent=styles.sasweb;
    import 'Hdr.CSS';
  end;
run;

ods html file='demo06_outfit6.html' style=demo06;
ods rtf file='demo06_outfit6.rtf' style=demo06;
ods pdf file='demo06_outfit6.pdf' style=demo06;

proc report data=prdsale nowd;
  title 'Product and Region Report';
  column product region,(actual predict)
         ('Total Sales' actual=acttot predict=predtot);
  define product / group style(column)=RowHeader;
  define region / across 'Region Sales';
  define actual / sum 'Actual' style(header)=AHDR;
  define predict/ sum 'Predicted' style(header)=PHDR;
  define acttot / sum 'Actual' style(header)=AHDR;
  define predtot / sum 'Predicted' style(header)=PHDR;
  rbreak after / summarize;
  compute product;
    if _break_ = '_RBREAK_' then do;
       product='Total';
       call define('product','style','style=RowHeader');
    end;
  endcomp;
run;

ods html close;
```

The code example shown above will also work for RTF and PDF destinations. If you want to see an HTMLCLASS example using AHDR and PHDR class selectors, or if you want to review the CSSSTYLE= option to accomplish the same results, those alternative methods are contained in the download file for this paper.

OUTFIT 7: CHANGING GRAPH OUTPUT WITH A STYLE TEMPLATE

Last, but not least, in this *haute couture* fashion show, is an example of how to change graph output using a style template. For this show, only a simple bar chart will be changed. All of the bars are the same height so that you can see the colors being used. In a bar chart, grouped colors are controlled by the **GraphData1**-**GraphData12** style elements. These style elements use the `gdata1-gdata12` style attributes from the **GraphColors** style element to control bar colors. The style template for this example will change bar colors for SGPLOT procedure output as well as for GCHART procedure output. (Note that the GCHART procedure output will only be affected by the style template if the GSTYLE system option is in effect.) The SGPLOT results are shown in Figure 13, and they were produced by the code immediately following the figure.

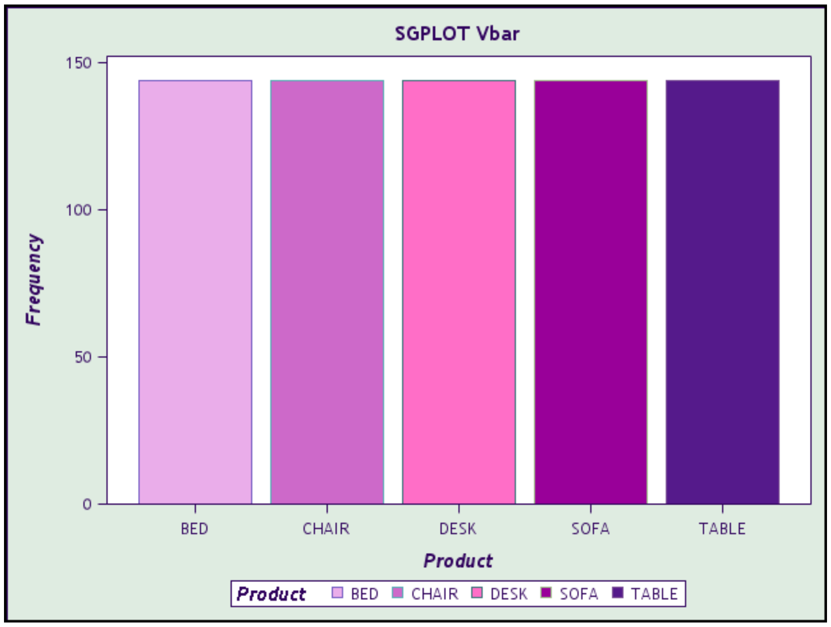


Figure 13: SGPLOT Program Results

```
ods path work.temp(update)
         sashelp.tmplmst(read);

proc template;
  define style styles.demo07;
  parent = styles.ocean;
  class GraphColors /
        'gdata1' = CXEAADEA
```

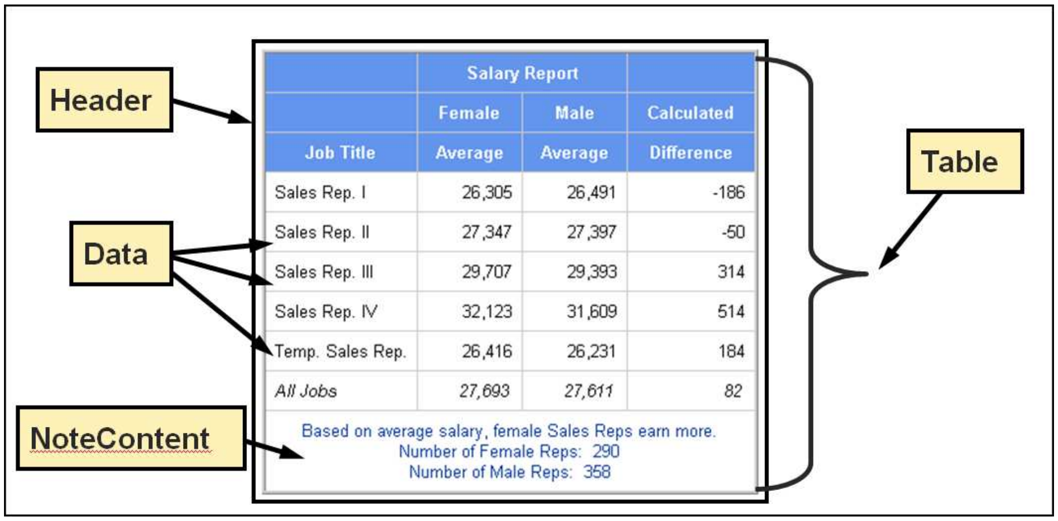
```
            'gdata2' = CXCD69C9
            'gdata3' = CXFF6EC7
            'gdata4' = CX990099
            'gdata5' = CX551A8B;
      end;
    run;

    ods listing style=demo07;
    proc sgplot data=prdsale;
      vbar product / group=product;
      title 'SGPLOT Vbar';
    run;
```

To effectively change the output of the new SG procedures, you need to consult the SAS/GRAPH documentation, which contains useful style element information for graph components. With classic SAS/GRAPH procedures, such as GCHART, GPLOT, and GMAP, you can continue to use your SAS/GRAPH syntax methods for changing output appearance, or you can use style template methods for changing the style. Keep in mind that, for classic SAS/GRAPH procedures, changes in SAS/GRAPH procedure and GOPTIONS, AXIS, SYMBOL, and PATTERN statements will override style settings from the style template. For the new SG procedures, changes in the SG procedure syntax will override what's in the style template.

HAUTE COUTURE DETAILS

Whether you're sewing a dress for a fashion show or designing a style template, you have to understand the basic components that you're working with. For example, no matter how fancy or plain a dress might be, all dresses have some basic components for which decisions have to be made: a dress has a neckline and/or collar treatment (PeterPan collar, mandarin collar, boat neckline, middy collar, cowl neckline, v-neck, sweetheart neckline, etc.); a dress has to accommodate arms (set-in sleeves, sleeveless, strapless, raglan sleeves, gathered sleeves, spaghetti straps, etc.); a dress has a fastening mechanism (zipper in the back, zipper at the side, wrap dress, buttons in the front, buttons in the back, hook and eyes, etc.); a dress has a bodice and waistline treatment (empire bodice and raised waist, darted bodice and gathered waist, princess style without a separate waist seam, tent dress with no waist, smocked bodice, etc.); and a dress has a skirt and hem treatment (full skirt, A-line skirt, pleated skirt, circle skirt, etc.).

In the same way, your SAS output has some basic components (or style elements) that you will work with over and over again as you design your style templates. Those style elements for tabular output are shown in Figure 14, and for graphic output, they are shown in Figure 15.



	Salary Report		
	Female	Male	Calculated
Job Title	Average	Average	Difference
Sales Rep. I	26,305	26,491	-186
Sales Rep. II	27,347	27,397	-50
Sales Rep. III	29,707	29,393	314
Sales Rep. IV	32,123	31,609	514
Temp. Sales Rep.	26,416	26,231	184
All Jobs	*27,693*	*27,611*	*82*

Based on average salary, female Sales Reps earn more.
Number of Female Reps: 290
Number of Male Reps: 358

Figure 14. Common Style Elements for Tabular Output

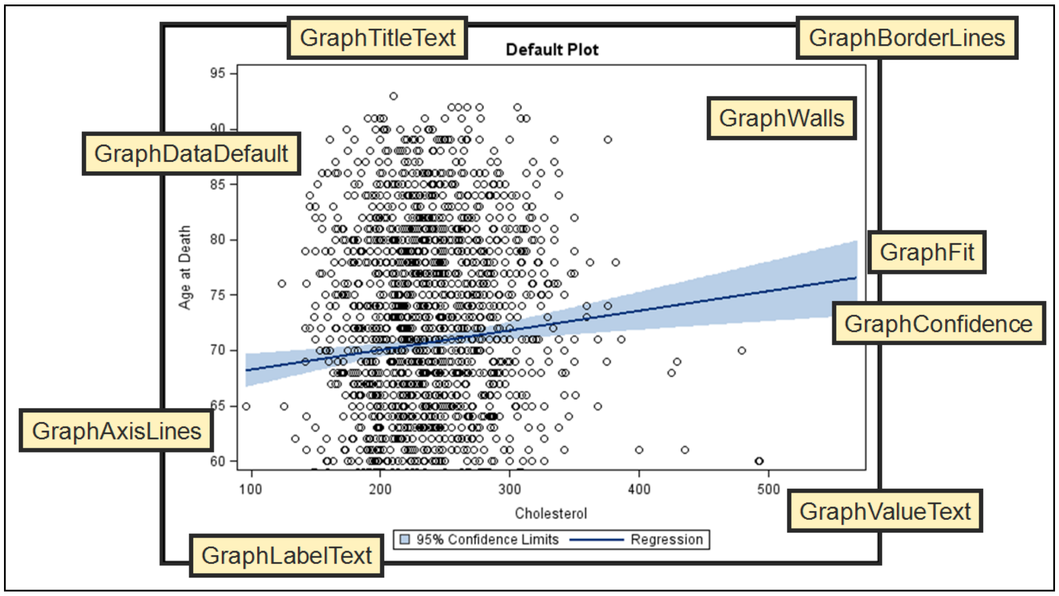
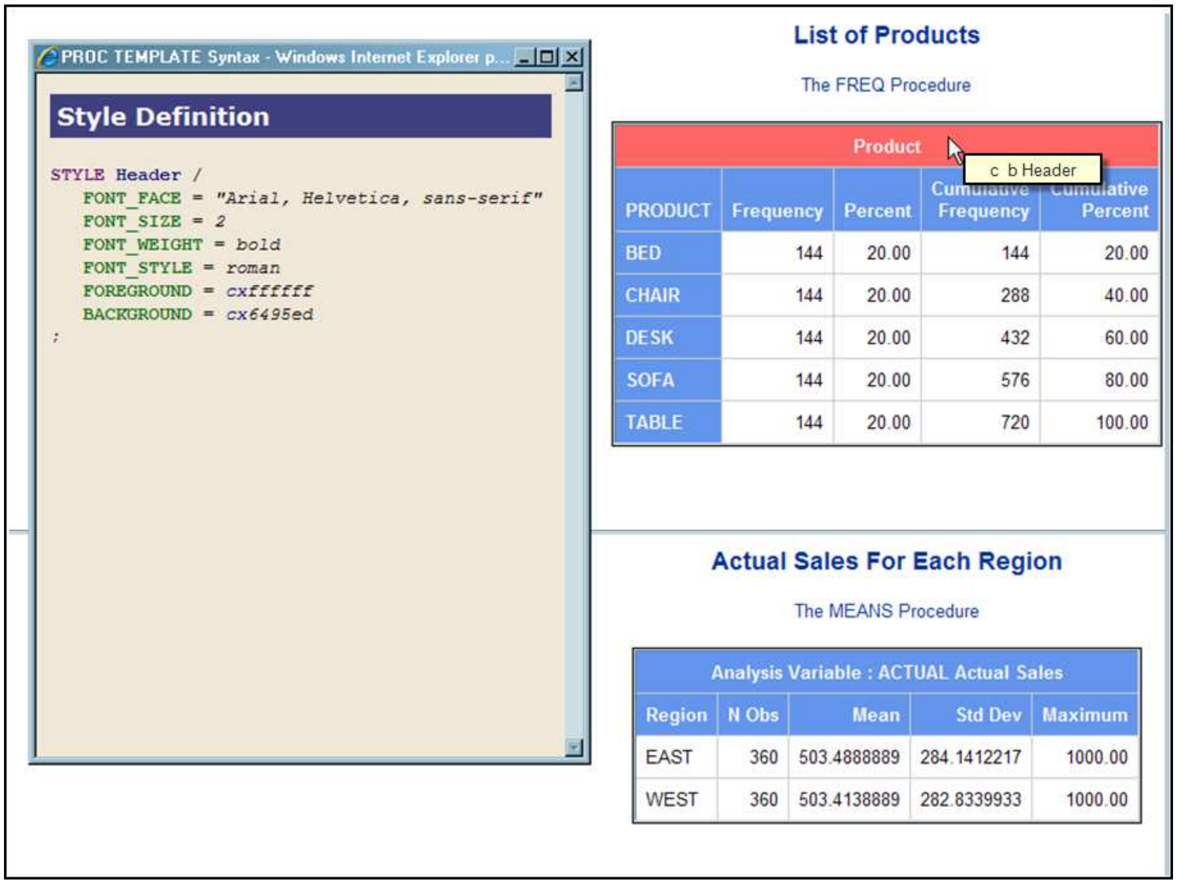


Figure 15. Common Style Elements shown for SGPLOT Procedure Output

Of course, there are a few more style elements (but not many) that you might need to work with, especially for tabular output. The **ByLine, Body, RowHeader**, **SystemTitle**, **SystemFooter**, **ProcTitle**, and **UserText** elements round out the list. And, it doesn't matter which SAS procedure you use. Just as a dress will always have some type of skirt, tabular procedure output will always have header area cells and data area cells that are styled by the **Header** and **Data** style elements, respectively. Do you absolutely need to know which style element the **Header** element inherits from in order to change the **Header** element? No! In SAS 9.2, with the new inheritance model in BASE.TEMPLATE.STYLE, all you need to do is use the CLASS statement (or STYLE statement) to change the visible style element. But, as shown in Outfit 2, where **Header** changed but **RowHeader** used the original template colors, knowing more about inheritance can be useful.

If you are uncertain about which style elements and attributes are used for your tabular output, you can use the TAGSETS.STYLE_POPUP (with Microsoft Internet Explorer) or TAGSETS.ODSSTYLE destination to help you determine exactly which style elements are used in your output and which style attributes are specified by those elements. The TAGSETS.STYLE_POPUP destination uses JavaScript and popup windows to display element and attribute information. The TAGSETS.ODSSTYLE destination resolves all inheritance into template code that is suitable for cutting and pasting. An example of TAGSETS.STYLE_POPUP output is shown in Figure 16, and an example of TAGSETS.ODSSTYLE output is shown in Figure 17. The complete code that produced the output shown in both of these figures is in the downloadable zip file that will be posted on the support.sas.com Web site. (See “Download Information” at the end of this paper.)



Style Definition

```
STYLE Header /
   FONT_FACE = "Arial, Helvetica, sans-serif"
   FONT_SIZE = 2
   FONT_WEIGHT = bold
   FONT_STYLE = roman
   FOREGROUND = cxffffff
   BACKGROUND = cx6495ed
;
```

List of Products

The FREQ Procedure

Product				
PRODUCT	Frequency	Percent	Cumulative Frequency	Cumulative Percent
BED	144	20.00	144	20.00
CHAIR	144	20.00	288	40.00
DESK	144	20.00	432	60.00
SOFA	144	20.00	576	80.00
TABLE	144	20.00	720	100.00

Actual Sales For Each Region

The MEANS Procedure

Analysis Variable : ACTUAL Actual Sales				
Region	N Obs	Mean	Std Dev	Maximum
EAST	360	503.4888889	284.1412217	1000.00
WEST	360	503.4138889	282.8339933	1000.00

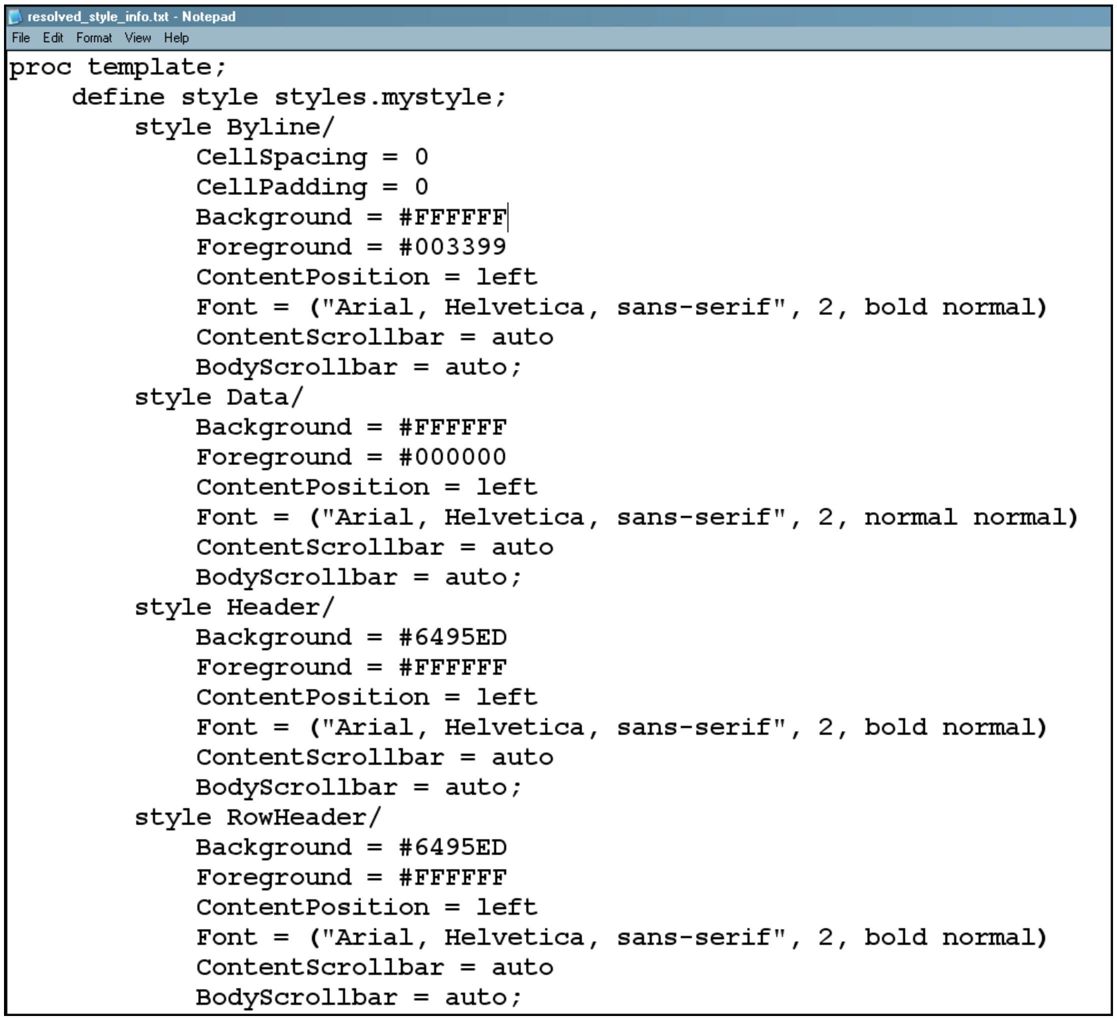
Figure 16. TAGSETS.STYLE_POPUP Output

To invoke the TAGSETS.STYLE_POPUP tagset, use the following code:

```
ods path sashelp.tmplmst(read);

ods tagsets.style_popup path='.' (url=none) file='spu_diagnose.html'
                        style=sasweb stylesheet='spu.css';
  . . . Your procedure code of interest . . .
ods _all_ close;
```

Note that popups must be enabled in your browser in order to see the popup window that displays the style elements and attributes. These elements and attributes can be used for PDF and RTF destinations as well; you use the HTML destination for diagnostic purposes only.

The TAGSETS.ODSSTYLE destination is also a good diagnostic tool, because it creates a file that can be used to design a new template from scratch (without any PARENT= statement). This file can also be used (in a verbose form) to see how inheritance was resolved across multiple parent templates or to cut and paste from when designing your own style templates (with or without a parent template in effect). You could change the STYLE statement in the ODSSTYLE output to a CLASS statement, or you can leave it as a STYLE statement. Since CLASS and STYLE are effectively the same if there isn't a parent style definition, either statement will work, because the template code generated by TAGSETS.ODSSTYLE does not have a PARENT statement. Partial output from TAGSETS.ODSSTYLE is shown in Figure 17.



```
proc template;
    define style styles.mystyle;
        style Byline/
            CellSpacing = 0
            CellPadding = 0
            Background = #FFFFFF
            Foreground = #003399
            ContentPosition = left
            Font = ("Arial, Helvetica, sans-serif", 2, bold normal)
            ContentScrollbar = auto
            BodyScrollbar = auto;
        style Data/
            Background = #FFFFFF
            Foreground = #000000
            ContentPosition = left
            Font = ("Arial, Helvetica, sans-serif", 2, normal normal)
            ContentScrollbar = auto
            BodyScrollbar = auto;
        style Header/
            Background = #6495ED
            Foreground = #FFFFFF
            ContentPosition = left
            Font = ("Arial, Helvetica, sans-serif", 2, bold normal)
            ContentScrollbar = auto
            BodyScrollbar = auto;
        style RowHeader/
            Background = #6495ED
            Foreground = #FFFFFF
            ContentPosition = left
            Font = ("Arial, Helvetica, sans-serif", 2, bold normal)
            ContentScrollbar = auto
            BodyScrollbar = auto;
```

Figure 17. Partial TAGSETS.ODSSTYLE Output

The output from TAGSETS.ODSSTYLE was rearranged for purposes of Figure 17 to show some of the most common elements at the top of the output file. In addition, a comment at the top of the output file is not shown in order to conserve space. Note how the TXT file created by TAGSETS.ODSSTYLE contains a syntactically correct PROC TEMPLATE step that defines a new style template called STYLES.MYSTYLE. Note also that there are no **colors**, **color_list**, or **fonts** style elements in this output file. That is because inheritance has been resolved into each STYLE statement, resulting in verbose, but very clear code.

```
ods path sashelp.tmplmst(read);

ods tagsets.odsstyle path='.' (url=none) file='dummy.html'
                     style=sasweb stylesheet='resolved_style_info.txt';
  . . . Your procedure code of interest . . .
ods _all_ close;
```

Whether you are just starting with style templates or you have worked with style templates in earlier versions of SAS, these two diagnostic tagset templates can help you understand the style elements and attributes that you need to use in your own style template design. In addition, the appendix to this paper contains a table (which is useful as a job aid) of common style attributes and their possible values. This table is an excerpt from the ODS documentation topic entitled, "Style Attributes and Their Values" (referenced in “Understanding the Pattern and Directions” above), which is a complete list of all the possible style attributes that you can specify in a style template. This documentation topic also contains information about which destinations support each attribute and the area of the output that is affected by the attribute. In addition, the file named COMMON_STYLE_ELEMENTS.TXT is in the zip file of programs that accompanies this paper.

CONCLUSION

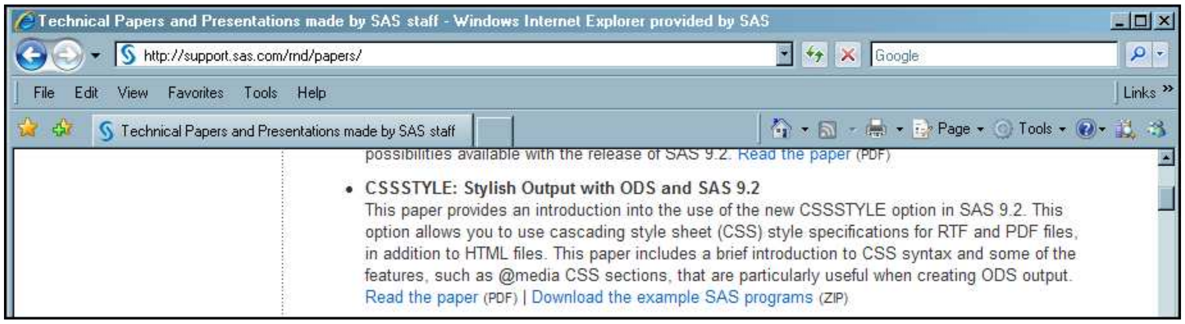
Learning to modify style templates has never been easier. Since the advent of SAS 9.2 and the introduction of the CLASS and IMPORT statements, you can fashion your own line of style templates complete with your company's preferred fonts and color palettes. Tracing inheritance issues is not as crucial when using SAS 9.2 as it was in SAS 8 or SAS 9.1.3, because the introduction of BASE.TEMPLATE.STYLE as the implicit style template parent simplifies the code that you have to write.

DOWNLOAD INFORMATION

Papers written by SAS staff for SAS Global Forum and other user group presentations are generally available for download from the Web site listed below. Papers presented at SAS Global Forum 2010 will be available shortly after the conference is over.

http://support.sas.com/rnd/papers

Search for your paper of interest by paper title (not by author name). Figure 18 shows the paper and download information for the author's CSSSTYLE paper from SAS Global Forum 2009.


Technical Papers and Presentations made by SAS staff - Windows Internet Explorer provided by SAS
http://support.sas.com/rnd/papers/
Google
File Edit View Favorites Tools Help
Links
Technical Papers and Presentations made by SAS staff
Page
Tools

possibilities available with the release of SAS 9.2. Read the paper (PDF)

- **CSSSTYLE: Stylish Output with ODS and SAS 9.2**
  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 includes a brief introduction to CSS syntax and some of the features, such as @media CSS sections, that are particularly useful when creating ODS output.
  Read the paper (PDF) | Download the example SAS programs (ZIP)

Figure 18. Support.sas.com site for paper and program download

REFERENCES AND RECOMMENDED READING

- Smith, Kevin D. 2006. "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.
- Smith, Kevin D. 2007. "The Output Delivery System (ODS) from Scratch." *Proceedings of the SAS Global Forum 2007 Conference*. Cary, NC. SAS Institute Inc. Available at http://support.sas.com/rnd/base/ods/scratch/ods-from-sc-paper.pdf.
- Cartier, J. 2002. "Visual Styles for V9 SAS® Output." *Proceedings of the Twenty-Seventh Annual SAS Users Group International Conference.* Cary, NC: SAS Institute Inc. Available at http://support.sas.com/rnd/datavisualization/papers/VisualStyles.pdf.
- Cartier, J. 2002. "The Basics of Creating Graphs with SAS/GRAPH® Software." *Proceedings of the Twenty-Seventh Annual SAS Users Group International Conference.* Cary, NC: SAS Institute Inc. Available at http://support.sas.com/rnd/datavisualization/papers/GraphBasics.pdf.
- Cartier, J. 2003. "It's All in the Presentation." *Proceedings of the Twenty-Eighth Annual SAS Users Group International Conference.* Cary, NC: SAS Institute Inc. Available at http://www2.sas.com/proceedings/sugi28/147-28.pdf.
- Zender, Cynthia L. 2009. "CSSSTYLE: Stylish Output with ODS and SAS® 9.2." *Proceedings of the SAS Global Forum 2009 Conference*. Cary, NC. SAS Institute Inc. Available at http://support.sas.com/resources/papers/proceedings09/014-2009.pdf.
- Zender, Cynthia L. 2009. "Tiptoe Through the Templates." *Proceedings of the SAS Global Forum 2009 Conference*. Cary, NC. SAS Institute Inc. Available at http://support.sas.com/resources/papers/proceedings09/227-2009.pdf.

- 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. 2003. "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.
- Heath, Dan. 2007. "New SAS/GRAPH® Procedures for Creating Statistical Graphics in Data Analysis." *Proceedings of the 2007 SAS Global Forum*. Cary, NC: SAS Institute Inc. Available at http://support.sas.com/rnd/datavisualization/papers/sgf2007/SGF2007-Proc.pdf.
- Heath, Dan. 2008. "Effective Graphics Made Simple Using SAS/GRAPH® SG Procedures." *Proceedings of the 2008 SAS Global Forum*. Cary, NC: SAS Institute Inc. Available at http://www2.sas.com/proceedings/forum2008/255-2008.pdf.
- Haworth, Lauren E., Cynthia L. Zender, and Michele M. Burlew. 2009. *Output Delivery System: The Basics and Beyond*. Cary, NC: SAS Institute Inc.

ACKNOWLEDGMENTS

As always, the author would like to thank David Kelley, Kevin Smith, Wayne Hester, Scott Huntley, Dan O'Connor, and Eric Gebhart for their efforts to make ODS as wonderful as it is (and for answering my questions about it). In addition, thanks are always due to Bari Lawhorn and Chevell Parker who possess awesome ODS problem solving skills. This paper has been made better because of Michele Ensor, Jane Stroupe, David Kelley, Kevin Smith, Bari Lawhorn, Chevell Parker, Lorilyn Russell, and Chris Riddiough, who reviewed it, and Jeanne Ferneyhough who edited it. I am very grateful to my grandmother and mother (both deceased), who taught me that if you have a sewing machine, a pattern, some fabric and patience, you can sew anything. The sewing machine didn't help me with ODS, but the patience did!

CONTACT INFORMATION

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

Cynthia Zender
SAS Campus Drive
SAS Institute Inc.
E-mail: Cynthia.Zender@sas.com

APPENDIX: COMMON STYLE ATTRIBUTES AND THEIR VALUES

Use With	Attribute	Possible Value
Table or cell style elements (such as Header RowHeader ByLine ProcTitle SystemTitle SystemFooter, etc.)	COLOR or FOREGROUND BACKGROUND or BACKGROUNDCOLOR	white, red, yellow, black, pink, purple cxFFFFFF, cxFF0000, cxFFFF00, cx000000 h000FF00, h07880FF, h0B480FF, h0000000
	FONT_FACE	Times "Times New Roman" Arial Helvetica Courier 'Courier New' "Arial, Helvetica, Helv"
	FONT_SIZE	5, 10 pt, 1 cm, 0.25 in
	FONT_STYLE	italic, roman
	FONT_WEIGHT	medium, bold, light
	FONT_WIDTH	compressed, narrow, wide
	FONT	('<sans-serif>,Arial, Helvetica, Helv', 2) ("Times, Times New Roman", 10pt, bold) ('Courier New, Courier', 14pt, bold italic)
	CELLWIDTH or WIDTH	150, 2 in, 5 cm, 20%
	CELLHEIGHT or HEIGHT	150, 2 in, 5 cm, 20%
	JUST	left, center, right or l, c, r
	VJUST	top, middle, bottom or t, m, b
	URL	'www.sas.com' "http://support.sas.com" 'C:\report_dir\Jan_Sales.html'
	PREIMAGE POSTIMAGE BACKGROUNDIMAGE	'C:\images\myimage.jpg' or "myimage.jpg" or 'http://www.srv.com/images/myimage.jpg'
	PREHTML POSTHTML	"<HR size=3>" '<a href="www.sas.com"> SAS Web Page</a>'
	PRETEXT POSTTEXT	'This is a draft.' "Continue"
Table style elements only (such as Table, Output or the "container" abstract style elements)	BORDERCOLOR	CX000000, black (or any color scheme as shown above)
	BORDERWIDTH BORDERSPACING PADDING	0, 7, .1 in, 1 cm
	RULES	ALL, COLS, GROUPS, NONE, ROWS
	FRAME	VOID, ABOVE, BELOW, BOX, HSIDES, LHS, RHS, VSIDES
	OUTPUTWIDTH or WIDTH	150, 3 in, 20 cm, 95%