Paper 091-31

Now–That's Your Style!!!!!

Chevell Parker, SAS Institute Inc., Cary, NC

ABSTRACT

This paper discusses how to use styles to enhance output or to overcome problems that are frequently encountered using the Output Delivery System (ODS). Solutions to these problems involve methods that use styles and tagsets in the TEMPLATE procedure, cascading style sheets (CSS), scripting, the Extensible Stylesheet Language (XSL), and more. Other topics that are addressed include handling page breaks in HTML; preventing truncated printed HTML output; adding headers, footers, page margins, and repeating headers; printing in landscape with the ODS HTML destination; and fine-tuning your printed HTML.

Other topics that are discussed are adding styles separately for the output that is viewed in the browser and the output that is printed; reducing the size of a table so that there's no need for scroll bars; adding scroll bars to the table when the output extends past the viewable screen; creating a perfect data grid that freezes column headers and row headers, sorts data, and describes data; and providing styles that look just like a Windows application (in which many users are familiar with). In addition, this paper addresses creating dynamic output; generating "that dream report" using styles; adding buttons that perform a variety of actions; interacting with other applications; e-mailing output using ODS; enhancing the Table of Contents; styling with XML; and a few other secrets.

INTRODUCTION

What is a style? *Merriam-Webster's Collegiate Dictionary* defines a style as "a distinctive manner of expression." In the past, style in reports was limited to black and white output, or, at best, green bars. Beginning with Version 7 of the SAS System and the introduction of ODS and the HTML destination, things have changed. ODS and HTML gave us colors, fonts, links, and more. Later, the RTF and PDF destinations became available. ODS was only the beginning of style in Base SAS software. Now, styles can be generated or modified several different ways with ODS.

In SAS 9.1, an internal CSS file provides formatting and is created by default by the ODS HTML destination. A CSS file can be either internal or external and is simply a text file with a style declaration that consists of the style property or selector and the value, separated by a colon. CSS files are reserved for markup languages and do not apply to destinations, such as RTF and PDF.

PROC TEMPLATE formats or modifies output that is generated with ODS. It creates a SAS member with a member type of ITEMSTOR. In SAS 9.2, PROC TEMPLATE includes a CSS parser, which allows CSS files that are generated by SAS, or files that follow the same structure as a CSS file that is generated by SAS, to be imported to generate a template definition.

The ODS ESCAPECHAR= statement allows you to format locally.

There are many more ways to create or modify style, such as scripting, using tagsets, and so on. This paper shows you how to use many of these methods. Note that the code produced in this paper was developed using the Microsoft Internet Explorer browser. Some of the code might not work in other browsers and is documented accordingly in the paper.

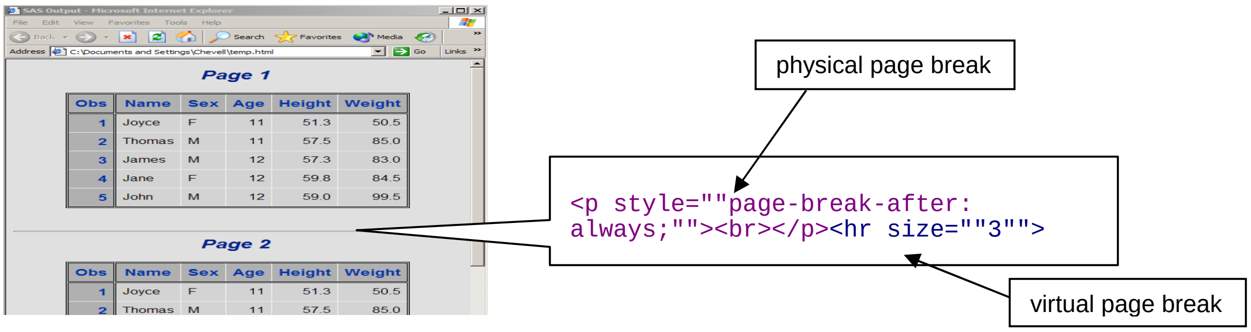
PRINTING

One of the largest sources of frustration when working with HTML is printing from a Web browser. This is mostly because HTML is considered to be a stream file, which is just one long page. In its infancy, HTML was designed to be viewed in a Web browser, not printed from it. However, users love to print from Web browsers, but there are many problems when printing, such as no true page breaks, truncated and clipped output, column headers that don't repeat on each page, and much more. With the help of the World Wide Web Consortium (W3C), there have been major improvements. This section discusses some of the printing problems and how to overcome them.

SUPPORTING PAGE BREAKS

HTML was initially meant to be viewed on the screen, not printed. Therefore, it does not have true page breaks. When printing from a Web browser, tables and graphs can get cut in the middle. When a table is cut in the middle, the table's observations continue printing on the next page without the table's header information.

Now, page breaks are supported by major Web browsers using the CSS properties **page-break-after** and **page-break-before**. The accepted values for these properties are **always**, **never**, **left**, and **right**. By default, the ODS HTML destination generates **page-break-after:always** within the paragraph tag after each output object. As a result, each output object is printed on a separate page; however, there are times when you do not want this to happen. The **PageBreakHTML=** attribute within PROC TEMPLATE removes all of the page breaks in the entire HTML document; individual page breaks can be selectively modified if you know the syntax to select a specific page break (Figure 1).


SAS Output - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address C:\Documents and Settings\Chevell\temp.html Go Links
physical page break
virtual page break

Page 1

Obs	Name	Sex	Age	Height	Weight
1	Joyce	F	11	51.3	50.5
2	Thomas	M	11	57.5	85.0
3	James	M	12	57.3	83.0
4	Jane	F	12	59.8	84.5
5	John	M	12	59.0	99.5

Page 2

Obs	Name	Sex	Age	Height	Weight
1	Joyce	F	11	51.3	50.5
2	Thomas	M	11	57.5	85.0

```
<p style=""page-break-after:
always;""><br></p><hr size=""3"">
```

Figure 1. Syntax for a Specific Page Break

REMOVING PAGE BREAKS

Using PROC TEMPLATE

You can remove page breaks from a file generated by the ODS HTML destination using PROC TEMPLATE or CSS. PROC TEMPLATE allows you to remove the CSS property **page-break-after** within the paragraph <P> tag and the horizontal rule <HR> tag after the paragraph tag. Once the page break is removed, you can modify the style manually, as desired. Specifically, you can use the **pageBreakHTML=** attribute with the value **_undef_** to remove the horizontal rule line (which you can see in Figure 1) and the CSS property (which is causing the physical page break).

Example Using the Style Template:

```
   proc template;
    define style styles.nobreak;
      parent=styles.default;
       style body from body /
         pagebreakhtml=_undef_;
    end;
   run;

   ods html file="temp.html" style=styles.nobreak;
   procedure1...
   procedure2...
   ods html close;
```

The next example modifies the pagebreak event of the **html4** tagset to remove page breaks, which the ODS HTML destination uses by default when generating HTML.

Example Using the Tagset Template:

```
   proc template;
     define tagset tagsets.nobreak;
       parent=tagsets.html4;
        define event pagebreak;
        end;
     end;
   run;
```

```
ods tagsets.nobreak file="temp.html";
procedure1...
procedure2...
ods tagsets.nobreak close;
```

Using CSS

The best way to remove the horizontal rule line and the physical page break using CSS is to add an ID or CLASS to the paragraph tag and the horizontal rule tag. This allows access to the specific paragraph tag and horizontal rule tag causing the virtual and physical page break. Using the CSS property **display:none** removes the horizontal rule line like it was never set. Therefore, the space that the horizontal rule line occupied is removed. Using the CSS property **visibility:hidden** hides the horizontal rule line and retains the space.

In the next example, PROC TEMPLATE is used to add an ID to the paragraph tag that is responsible for the virtual and physical page break. Then, the ID is added to the **headtext=** option, which adds the ID selector between the <head> and </head> tags which acts as a global style sheet.

```
proc template;
 edit styles.default as styles.test;
  edit html /
   'pagebreakline' =
   "<p id=""pagebreak"" style=""page-break-after:always;""><br></p><hr
id=""pagebreak"" size=""3"">" ;
  end;
run;

ods html file="temp.html" style=styles.test headtext="<style> #pagebreak
{display:none}</style>";
proc print data=sashelp.class;
run;

proc print data=sashelp.class;
run;

ods html close;
```

Using Scripting

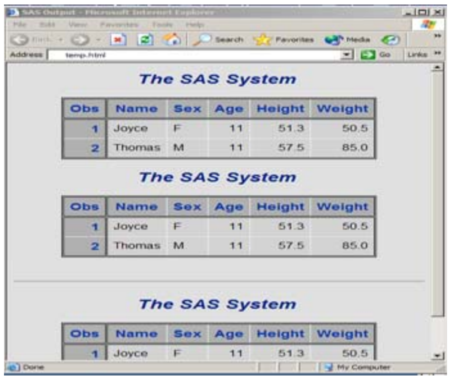
You can remove page breaks within a script by setting the **pageBreakAfter** property of the object to null. In the next example, the JavaScript variable **coll** locates the first paragraph tag and the **for** loop cycles through all of the paragraph tags and sets **pageBreakAfter** to null. This script can be added to PROC TEMPLATE by using style elements such as **StartUpFunction**, or externally by using the **headtext=** option and including an external link to the JavaScript.

```
var coll = document.all.tags("P");
for (i=0; i<coll.length; i++) {
    coll(i).style.pageBreakAfter = "";
}

ods html body='temp.html' headtext=
       '<SCRIPT LANGUAGE="JavaScript"
       TYPE="text/JavaScript" SRC="c:\test.js"></SCRIPT>' ;

 proc print data=sashelp.class;
 run;

 ods html close;
```

MODIFYING PAGE BREAKS

The page break is handled on a document level with styles in PROC TEMPLATE; therefore, it cannot be toggled between tables--it is either off or on. You can turn a page break on f manually with the **text=** option and the appropriate string responsible for page break. The **text=** option is new to the ODS HTML destination in SAS 9.1. Without adding this string with the **text=** option, all of the tables break at the printer's discretion. The next example creates a macro variable with the page break information and applies it after the first two tables. The output of this example is shown in Figure 2.

```
proc template;
 define style styles.test;
  parent=styles.default;
   style body from body /
   pagebreakhtml=_undef_;
  end;
run;

%let pbreak=%nrstr(<p style="" page-break-after:always""><br></p><hr/>);

ods html file="temp.html" style=styles.test;
 proc print data=sashelp.class(obs=2);
 run;
 proc print data=sashelp.class(obs=2);
 run;

ods html text="&pbreak";
 proc print data=sashelp.class(obs=2);
 run;

ods html close;
```



The SAS System

Obs	Name	Sex	Age	Height	Weight
1	Joyce	F	11	51.3	50.5
2	Thomas	M	11	57.5	85.0

The SAS System

Obs	Name	Sex	Age	Height	Weight
1	Joyce	F	11	51.3	50.5
2	Thomas	M	11	57.5	85.0

The SAS System

Obs	Name	Sex	Age	Height	Weight
1	Joyce	F	11	51.3	50.5

Figure 2. Page Break After the First Two Tables

SCRIPTING BY INCREMENTING

The next example extends the previous example that used the **for** loop by changing the loop's increment. Instead of incrementing the loop by the default value of 1 (i++), the loop is incremented by 2, which allows two tables on a page, as long as the two tables fit on a page.

```
proc template;
 define style styles.controlbreak;
  parent=styles.default;
   style startupfunction /
    tagattr='var coll = document.all.tags("P");
     for (i=0; i<coll.length; i+=2) {
      coll(i).style.pagebreakafter = "";
    }';
 end;
run;

ods html file="temp.html" style=styles.controlbreak;
proc print data=sashelp.class(obs=2);
run;
proc print data=sashelp.class(obs=2);
run;
proc print data=sashelp.class(obs=2);
run;
proc print data=sashelp.class(obs=2);
run;
ods html close;
```

CHANGING PAGE SETUP

Using ActiveX Controls

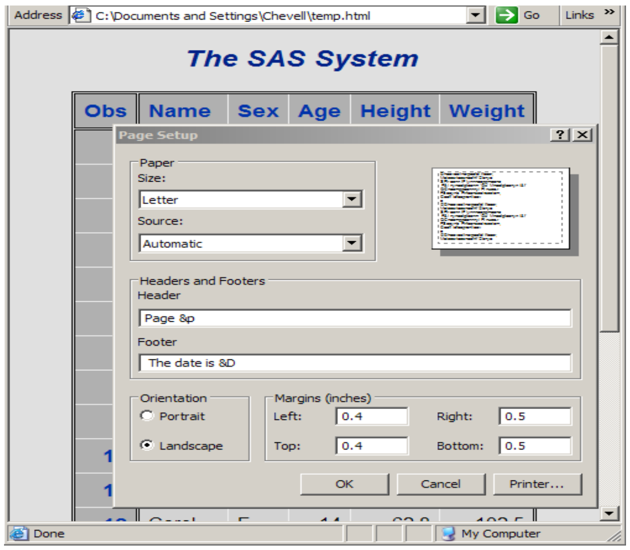
Page setup items can be modified using ActiveX controls. In the next example, an ActiveX control modifies the page orientation, headers, footers, and page margins from ODS. Web browsers have limited functionality of controlling page setup items programmatically, so this is one way to have the printed output exactly the way you want it. To format the headers and footers, see Table 1 below with the various codes that you can add to the header and footer section.

The example uses a tagset. (The tagset is designed for Microsoft Internet Explorer.) The tagset creates 5 macro variables with the **mvar** statement. The macros pass parameters for the header, the footer, orientation and the margins. The **object** tag with an ID and the JavaScript function call are appended to the event **doc_body** of the **html4** tagset. The **codebase=** option points to the cabinet (CAB) file that contains the ActiveX control. If you prefer not to use Activex controls, you can use the new printing dynamics of Microsoft Internet Explorer to create the behavior of the script.

```
proc template;
 define tagset tagsets.print;
  parent=tagsets.html4;
   mvar test_header test_footer test_orientation test_lmargin test_rmargin;
    define event doc_head;
     start:
       put "<head>" NL;
       put VALUE NL;
     finish:
       put "</head>" NL;
     end;
     define event doc_body;
       put "<body onload=""viewinit()""";
        put "onunload=""shutdown()""";
        put "bgproperties=""fixed""" / WATERMARK;
        putq "class=" HTMLCLASS;
        putq "background=" BACKGROUNDIMAGE;
        trigger style_inline;
        put ">" NL;
        trigger pre_post;
        put NL;
        trigger ie_check;
        put "<object id=""sugi"" style=""display:none"" viewastext" NL;
        put "classid=""clsid:1663ed61-23eb-11d2-b92f-008048fdd814""" NL;
        put "codebase=""http://www.meadroid.com/scriptx/ScriptX.cab#Version=6,1,432,1"">" NL;
        put "</object>" NL;
        put NL;
        put "<script defer>" NL;
        put "function viewinit() {"  NL;
        put "if (!sugi.object) {" NL;
        put "return" NL;
```

```
            put "} else {" NL;
            put "sugi.printing.header=" test_header NL;
            put "sugi.printing.footer=" test_footer NL;
            put "sugi.printing.portrait=" test_orientation NL;
            put "sugi.printing.leftmargin=" test_lmargin NL;
            put "sugi.printing.rightmargin=" test_rmargin NL;
            put "}" NL;
            put "}" NL;
            put "</script>" NL;
           finish:
            trigger pre_post;
            put "</body>" NL;
       end;
   end;
   run;

   %let test_header=%nrstr("Page &p");
   %let test_footer=%nrstr("The date is &D");
   %let test_orientation=false;
   %let test_lmargin=.4;
   %let test_rmargin=.5;

   ods tagsets.print file="temp.html";
   proc print data=sashelp.class;
   run;
   ods tagsets.print close;
```

Figure 3 shows the modified page setup used in the tagset.



C:\Documents and Settings\Chevell\temp.html

The SAS System

Obs Name Sex Age Height Weight

Page Setup

Paper
Size: Letter
Source: Automatic

Headers and Footers
Header
Page &p
Footer
The date is &D

Orientation: Portrait [✓ Landscape]

Margins (inches)
Left: 0.4 Right: 0.5
Top: 0.4 Bottom: 0.5

Figure 3. Page Setup Window for Printed HTML Files

Customizing Header and Footer Information
Header and footer information can be changed by using the Page Setup window (shown in Figure 3).

Usually, headers and footers are specified by default (such as **&w**, which specifies the window's title, **&p**, which specifies the page number, and **&u**, which specifies the address of the page). To change header and footer information, use the following codes.

Description	Value
Window title	&w
Page address (URL)	&u
Date in short format	&d
Date in long format	&D
Time in the format specified by Regional settings from the Control Panel	&t
Time in 24-hour format	&T
Current page number	&p
Total number of pages	&P
Right-aligned text (following &b)	&b
Centered text (between &b&b)	&b&b
A single ampersand (&)	&&

Table 1. Codes for Changing Headers and Footers in Printed HTML Files

If you are using the default header, then you can change the window title and the printed header title using the **title=** ODS HTML option, which allows you to specify a title for the window. This title overrides the value supplied within the <title> tag in the header of the HTML file, and is used as the printed header title.

```
ods html file="temp.html" (title="This appears as the window and the header title");
```

Changing Page Orientation

For some browsers, page orientation can be set using the CSS **@Page** rule. This rule allows you to set the orientation of the output, the page size, the margins, and a few other properties. The **@Page** rule is set to auto by default; other values are portrait and landscape. Currently, Opera is the only browser that supports the **@Page** rule.

```
<style>@Page {size:landscape}</style>
```

If you need to change the orientation of your output to landscape and you are not using the Opera browser, then you can use the **orientation=landscape** option with a destination such as the listing or the RTF destination. The landscaped output can then be specified as an alternate file to be printed which is discussed in the next section.

```
options orientation=landscape;

ods rtf file="c:\temp.rtf";
ods html file="c:\temp.html"
headtext='<LINK media="print" rel="alternate" href="temp.rtf">';

proc print data=sashelp.class;
run;

ods _all_ close;
```

SPECIFYING ALTERNATE PRINT

You can specify that an alternate file be printed when the user prints a file. For example, if the user clicks **Print**, you can specify that a different file be printed in place of the HTML file that the user is currently viewing on the browser. This removes the need for the user to select a printed version of what he or she is viewing.

To do this, use the **rel=** attribute with the **alternate** value, and the **media=** attribute with the **print** value. You can place these attributes and the **LINK** tag in the **headtext=** option so that it is included in the header of the HTML file. In the next example, when the user clicks **Print**, the file **temp.rtf** is printed in place of the HTML file, which prints features such as page numbers that are not printed with the HTML file.

```
ods rtf file="c:\temp.rtf";
ods html file="c:\temp.html"
headtext='<LINK media="print" rel="alternate" href="temp.rtf">';

proc print data=sashelp.class;
run;

ods _all_ close;
```

REPEATING COLUMN HEADERS IN A TABLE

Another problem when printing an HTML file is that a table's column headers do not repeat if a table spans more than a single page. This is also a problem when viewing the table online. HTML is considered to be a stream file, which is just one long page; therefore, the output that is printed is the output that is displayed, and column headers are not repeated online. You can use the CSS **display** property in either the **headtext=** option or in the CSS file with the value **table-header-group**. The **display** property repeats column headers on every page that is printed. This property does not repeat titles and footnotes on every page. Some browsers, such as Firefox, repeat column headers by default when the HTML file contains the <thead> section with <th> tags.

```
ods html file="temp.html"
headtext"<style> thead {display:table-header-group}</style>";
proc print data=sashelp.class;
run;
ods _all_ close;
```

PREVENTING TRUNCATED OUTPUT

When output on a Web browser is wider than the viewable screen, a scroll bar is added that enables you to view all of the output. However, when you print this output, it is truncated. You can scale the output using the **@media** rule or the **media** attribute along with **zoom** or **font-size** CSS style properties. By using one of these, viewable output is formatted differently than printed output. With the **@media** rule or **media** attribute, you must specify a media type.

Media Types
all
Braille
embossed
handheld
print
projection
screen
tty

The media type can be specified with the **@media** rule in the **style** tag or with the **media** attribute in the **style** object, or the style or link tags.

Example Using the **@media** Rule:

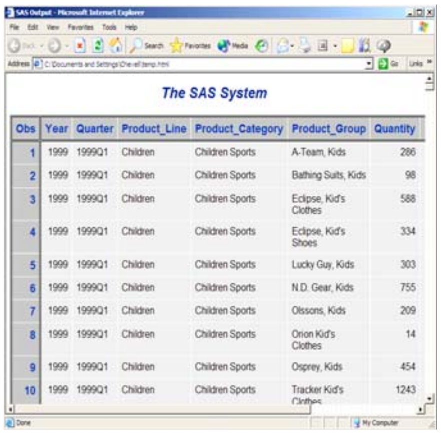
```
ods html file="temp.html"
headtext="<style> @media print {thead, tbody {zoom:50%}}</style>";
proc print data=sashelp.orsales;
run;
ods html close;
```

Example Using the **media** Attribute:

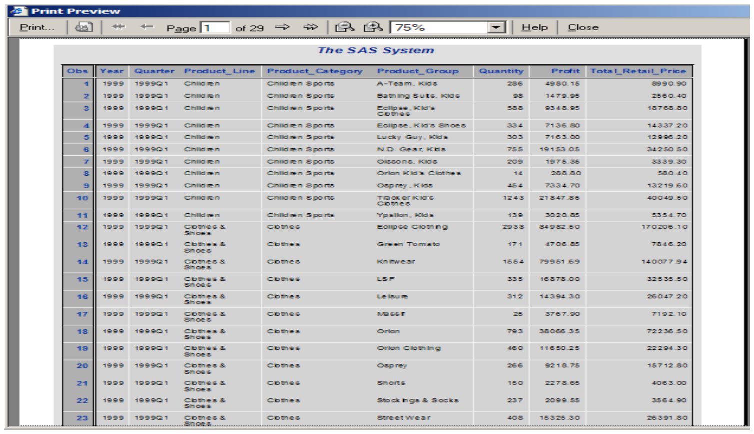
```
ods html file="temp.html"
headtext="<style media=""screen, print"" type=""text/css"">
  tbody,thead {zoom:50%}</style>";
```

```
   proc print data=sashelp.class;
   run;
   ods html close;
```

In the first screen in Figure 4, the “Quantity” column is the last variable of the table that is displayed. Thirty percent of the output is outside of the viewable screen. Because we are printing portrait, this 30% of the output will be clipped or truncated when printed. To prevent the output from being truncated, it is scaled using the **zoom** property. The scaled output is shown in the second screen in Figure 4.


The SAS System

Obs	Year	Quarter	Product_Line	Product_Category	Product_Group	Quantity
1	1999	1999Q1	Children	Children Sports	A-Team, Kids	286
2	1999	1999Q1	Children	Children Sports	Bathing Suits, Kids	98
3	1999	1999Q1	Children	Children Sports	Eclipse, Kid's Clothes	588
4	1999	1999Q1	Children	Children Sports	Eclipse, Kid's Shoes	334
5	1999	1999Q1	Children	Children Sports	Lucky Guy, Kids	303
6	1999	1999Q1	Children	Children Sports	N.D. Gear, Kids	755
7	1999	1999Q1	Children	Children Sports	Olssons, Kids	209
8	1999	1999Q1	Children	Children Sports	Orion Kid's Clothes	14
9	1999	1999Q1	Children	Children Sports	Osprey, Kids	454
10	1999	1999Q1	Children	Children Sports	Tracker Kid's Clothes	1243



The SAS System

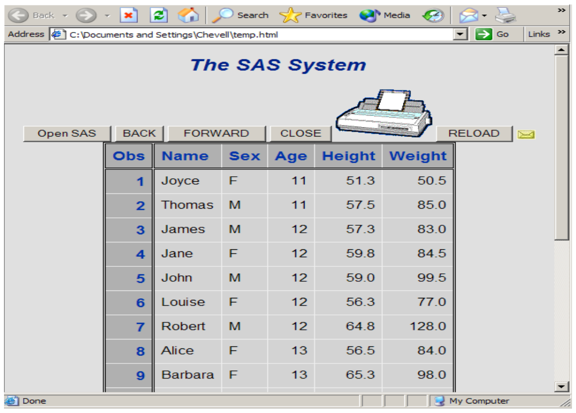
Obs	Year	Quarter	Product_Line	Product_Category	Product_Group	Quantity	Profit	Total_Retail_Price
1	1999	1999Q1	Children	Children Sports	A-Team, Kids	286	4980.15	8990.90
2	1999	1999Q1	Children	Children Sports	Bathing Suits, Kids	98	1479.95	2560.40
3	1999	1999Q1	Children	Children Sports	Eclipse, Kid's Clothes	588	9348.95	18768.80
4	1999	1999Q1	Children	Children Sports	Eclipse, Kid's Shoes	334	7136.80	14337.20
5	1999	1999Q1	Children	Children Sports	Lucky Guy, Kids	303	7163.00	12996.20
6	1999	1999Q1	Children	Children Sports	N.D. Gear, Kids	755	19153.05	34250.50
7	1999	1999Q1	Children	Children Sports	Olssons, Kids	209	1975.35	3339.30
8	1999	1999Q1	Children	Children Sports	Orion Kid's Clothes	14	288.80	580.40
9	1999	1999Q1	Children	Children Sports	Osprey, Kids	454	7334.70	13219.60
10	1999	1999Q1	Children	Children Sports	Tracker Kid's Clothes	1243	21847.85	40049.50
11	1999	1999Q1	Children	Children Sports	Ypsilon, Kids	139	3020.85	5354.70
12	1999	1999Q1	Clothes & Shoes	Clothes	Eclipse Clothing	2938	84982.50	170206.10
13	1999	1999Q1	Clothes & Shoes	Clothes	Green Tomato	171	4706.85	7846.20
14	1999	1999Q1	Clothes & Shoes	Clothes	Knitwear	1554	79951.69	140077.94
15	1999	1999Q1	Clothes & Shoes	Clothes	LSF	335	16878.00	32535.50
16	1999	1999Q1	Clothes & Shoes	Clothes	Leisure	312	14394.30	26047.20
17	1999	1999Q1	Clothes & Shoes	Clothes	Massif	25	3767.90	7192.10
18	1999	1999Q1	Clothes & Shoes	Clothes	Orion	793	38066.35	72236.50
19	1999	1999Q1	Clothes & Shoes	Clothes	Orion Clothing	460	11650.25	22294.30
20	1999	1999Q1	Clothes & Shoes	Clothes	Osprey	266	9218.75	15712.80
21	1999	1999Q1	Clothes & Shoes	Clothes	Shorts	150	2278.65	4063.00
22	1999	1999Q1	Clothes & Shoes	Clothes	Stockings & Socks	237	2099.55	3564.90
23	1999	1999Q1	Clothes & Shoes	Clothes	Street Wear	408	15325.30	26391.80

Figure 4. Output Before Scaling and After Scaling

ADDING BUTTONS

You can add buttons to a Web page (Figure 5), which enables users to perform actions without leaving the page. The JavaScript **window.print()** method invokes the Print dialog box and enables the user to print. The **history.forward()** and **history.back()** methods enable the user to go to the last page or the previous page visited. You can set up a contact button that sends an e-mail. Images can be added to buttons by changing the **type** from **button** to **image**, and then specifying the location of the image with the **src=** option. The next example creates several buttons and enables printing.

```
   proc template;
    define style styles.button;
     parent=styles.default;
     style table from output /
      prehtml='<div text-align:center>
       <input type="button" value="Open SAS" onclick=window.open("http://www.sas.com")>
       <input type="button" value="BACK" onclick="history.back()">
       <input type="button" value="FORWARD" onclick="history.forward()">
       <input type="button" value="CLOSE" onclick="window.close()">
       <input type="image" src="printer.gif" onclick="window.print()">
       <input type="button" value="RELOAD" onclick="location.reload()">
       <input type="image" src="mailto.jpg" onclick="location.href=''mailto:joe@sas.com''">
      </div>' ;
    end;
   run;

   ods html body='temp.html' style=styles.button;
    proc print data=sashelp.class;
    run;
   ods html close;
```



The SAS System

Obs	Name	Sex	Age	Height	Weight
1	Joyce	F	11	51.3	50.5
2	Thomas	M	11	57.5	85.0
3	James	M	12	57.3	83.0
4	Jane	F	12	59.8	84.5
5	John	M	12	59.0	99.5
6	Louise	F	12	56.3	77.0
7	Robert	M	12	64.8	128.0
8	Alice	F	13	56.5	84.0
9	Barbara	F	13	65.3	98.0

Figure 5. Web Page with Added Buttons

ACCESSING WebBrowser CONTROLS

Accessing WebBrowser controls with Microsoft Internet Explorer allows you to do things such as print from a button without invoking the Print dialog box, access the Page Setup window, access the Print Preview window, and more. Access to controls is accomplished using the **ExecWB** method and parameters that are passed to the method. Or, you can use the numeric representation, such as **ExecWB(6,-1)**. Table 2 is a partial list of supported commands for the **ExecWB** method. An example that uses the **ExecWB** method follows the list.

Command Name	Command Value
OLECMDID_OPEN	1
OLECMDID_NEW	2
OLECMDID_SAVE	3
OLECMDID_SAVEAS	4
OLECMDID_PRINT	6
OLECMDID_PRINTPREVIEW	7
OLECMDID_PAGESETUP	8
OLECMDID_PROPERTIES	10

Table 2. Partial List of Supported Commands

```
proc template;
 define style styles.test;
  parent=styles.default;
   style table from table /
    prehtml='<div style="text-align:center">
     <input onclick="test.ExecWB(6,-1)" value="No Prompt" type="button">
     <input onclick="test.ExecWB(4,1)" value="Save As" type="button">
     <input onclick="test.ExecWB(6,1)" value="Print Prompt" type="button">
     <input onclick="test.ExecWB(8,1)" value="Page Setup" type="button">
    </div>';
 end;
run;

ods html file='temp.html' style=styles.test
headtext='<OBJECT ID="test" CLASSID="CLSID:8856F961-340A-11D0-A96B-00C04FD705A2">
</OBJECT>';

proc print data=sashelp.class;
run;
ods html close;
```

PERSONALIZING THE VIEWING EXPERIENCE

PREVENTING SCROLL BARS

When output on a Web browser is wider than the viewable screen, a scroll bar is added that enables you to view all of the output. Ideally, the user should be able to see all of the output without having to scroll. You can use the **@media** rule to scale the output so that it fits in the viewable screen. Or, in PROC TEMPLATE, you can modify the Container style element and the **font_size=** attribute so that the output fits in the viewable screen.

Example Using the **@media** Rule:

```
ods html file="temp.html"
headtext="<style> @media all {tbody,thead {zoom: 80%}} </style>";
proc print data=sashelp.class;
run;
ods html close;
```

Example Using PROC TEMPLATE:

```
proc template;
 edit styles.default as styles.test;
  style container /
   font_size=1;
 end;
run;

ods html file="temp.html" style=styles.test;
proc print data=sashelp.class;
run;
ods html close;
```

CHANGING DISPLAY BASED ON SCREEN RESOLUTION

Many Web pages are generated based on a screen resolution of 800×600 with no variance; however, it is possible to build output based on screen resolution. For example, a Web page that is based on a screen resolution of 1024×768 might make some users have to scroll to see all of the data. If you were to build output based on screen resolution, you could control how the output looks on every user's screen. Using the JavaScript properties **screen.width** and **screen.height**, you can determine the screen resolution of the user's system. Based on the user's screen resolution, you can automatically build how you want the output to look. In the next example, the JavaScript **document.write( )** function writes out a style based on the screen resolution information.

Example Scaling Output Writing a Style:

```
ods html file="temp.html"
headtext='<SCRIPT LANGUAGE="JavaScript">
 if ((screen.width<=800) && (screen.height<=600)) {
    document.write("<style> body {zoom:80%}</style>")
}
</SCRIPT>';

proc print data=sashelp.class;
run;
ods html close;
```

Example Linking to a CSS File:

```
ods html file="c:\temp.html"
headtext='<SCRIPT LANGUAGE="JavaScript">
 if ((screen.width<=800) && (screen.height<=600)) {
    document.write(''<link rel="stylesheet" href="temp.css">'')
}
</SCRIPT>';
```

```
proc print data=sashelp.class;
run;
ods html close;
```

FREEZING A TABLE'S COLUMN AND ROW HEADERS

Some of the same problems when printing output exist when viewing output. HTML is considered to be a stream file, which is just one long page. If a Web page has a long and wide table, the user will forget, while scrolling, what the table's column headers and row headers are.

To solve this problem, you can freeze a table's column and row headers. Within a table, output can be dynamically positioned by performing the following steps.

1. Start with PROC TEMPLATE.
2. Modify the **style element** table.
3. Add the **prehtml=** attribute.
4. Within the **prehtml** attribute, use the <div> tag to create a container around the table.
5. Within the <div> tag, use the **overflow:auto** property so that if the contents of the container are longer or wider than the arguments specified by the **width** and **height** properties, either a vertical or horizontal scroll bar is added.
6. Within the <div> tag, use the **id=** attribute to freeze the table's column headers and row headers. The ID specified in the <div> tag allows the column headers and the row headers to access the dimensions of the container.
7. Next, use the **header** and **rowheader** style elements with the top and left CSS style properties along with the **expression** function. In the **expression** function, two separate methods are needed to calculate the location of the column headers and row headers.
   a. The **scrollTop** method determines the top-most part of the output, which freezes the column headers.
   b. The **scrollLeft** method determines the left-most part of the output, which freezes the row headers.
8. Finally, the **z-index** property handles any column overlap.

The following code shows an example of how to freeze row headers and column headers.

```
proc template;
  define style styles.test;
    parent=styles.default;
      style table from table /
        outputwidth=99%
        prehtml='<div style="overflow:auto; width:45%;height:300px" id="xxx">'
        posthtml="</div>";
      style header from header /
        htmlstyle='z-index:20; position:relative;
                  top:expression(document.getElementById("xxx").scrollTop)';
      style rowheader from rowheader /
        htmlstyle='position:relative;
                  left:expression(Container=xxx.scrollLeft)';
    end;
run;
ods html file="temp.html" style=styles.test ;

proc print data=sashelp.shoes(obs=200) noobs;
id region /style(header)={htmlclass="rowheader header" htmlstyle="z-index:30"};
run;

ods html close;
```

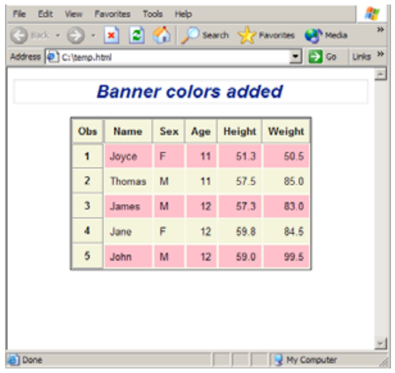
The previous code works only in the Microsoft Internet Explorer browser. The next example shows a workaround so that the code works in a Mozilla browser.

```
proc template;
  define tagset tagsets.test;
    parent=tagsets.html4;
      define event table_body;
        put "<tbody";
        putq "id=""test""";
        put ">" NL;
      finish:
        put "</tbody>" NL;
    end;
 end;
run;

ods tagsets.test file="c:\mozilla.html"
headtext="<style> #test {overflow:Auto;width:50%;height:100} </style>";

proc print data=sashelp.class;
run;
ods _all_ close;
```

INTRODUCTION TO THE TABLE EDITOR

ADDING STYLE

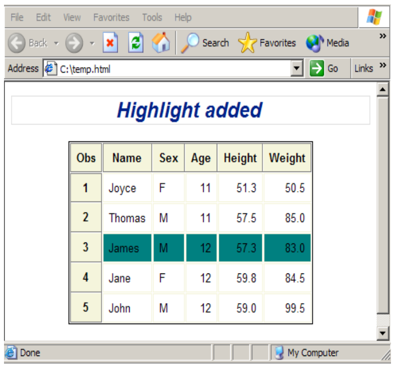
The Table Editor enables you to add style to your output. You can use the **banner_color_even=** option and the **banner_color_odd=** option to add background colors to odd and even rows of data that don't currently have background colors. With the **highlight_color**= option, you can select a specific row or column and highlight it. If the selected row or column has a banner color, highlighting takes precedence over the banner color. The **image_path** and **image_just**= options can be used to include an image and justify it. The **background_color=** option adds a background color to the body file. The **background_image=** option adds a background image to the body file. Using the **scrollbar_color=** option modifies the color of the scrollbar. Code examples and resulting output are shown next.



Banner colors added

Obs	Name	Sex	Age	Height	Weight
1	Joyce	F	11	51.3	50.5
2	Thomas	M	11	57.5	85.0
3	James	M	12	57.3	83.0
4	Jane	F	12	59.8	84.5
5	John	M	12	59.0	99.5

```
options(banner_color_even="beige"
        banner_color_odd="pink")
```



Highlight added

Obs	Name	Sex	Age	Height	Weight
1	Joyce	F	11	51.3	50.5
2	Thomas	M	11	57.5	85.0
3	James	M	12	57.3	83.0
4	Jane	F	12	59.8	84.5
5	John	M	12	59.0	99.5

```
options(highlight_color="teal")
```

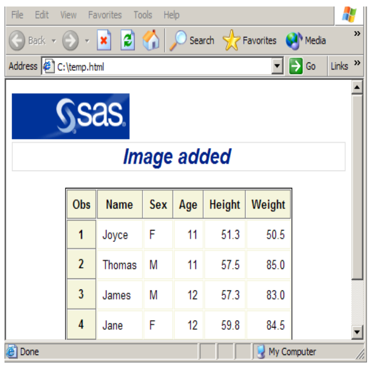


Image added

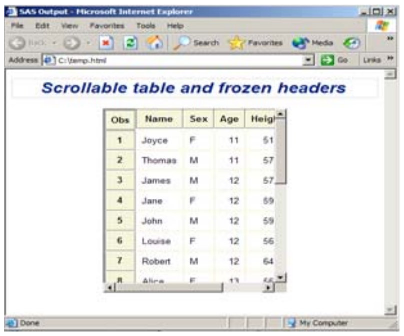
Obs	Name	Sex	Age	Height	Weight
1	Joyce	F	11	51.3	50.5
2	Thomas	M	11	57.5	85.0
3	James	M	12	57.3	83.0
4	Jane	F	12	59.8	84.5

```
options(image_path="c:\logo.gif")
```

Figure 6. Output Showing Added Styles

DYNAMICALLY POSITIONING AND FREEZING COLUMNS

When building a table, column and row headers can be frozen. The **frozen_headers** and **frozen_rowheaders** options are used with the **pagewidth=** and **pageheight=** options to freeze column and row headers. The **pagewidth** and **pageheight** options determine the width and the height of the page and properly display the table based on their values. In addition, when the page's maximum width and height are reached, scroll bars are added.


SAS Output - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address C:\temp.html
Go Links
Done
My Computer

Scrollable table and frozen headers

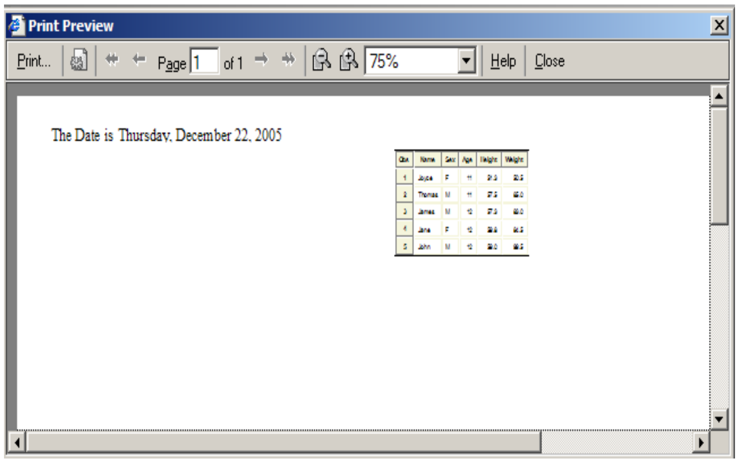
Obs	Name	Sex	Age	Heigh
1	Joyce	F	11	51
2	Thomas	M	11	57
3	James	M	12	57
4	Jane	F	12	59
5	John	M	12	59
6	Louise	F	12	56
7	Robert	M	12	64
8	Alice	F	13	56

```
options(pagewidth="50%" pageheight="300px" frozen_headers="yes"; frozen_rowheaders="yes")
```

Figure 7. Output Showing Added Scroll Bars and Frozen Column and Row Headers

Changing Page Setup and Printing Preferences

The **print_header**= and **print_footer**= options specify text to be printed in the header and footer of the printed output. Table 1 lists valid codes for changing the header and footer. The **orientation**= option specifies the orientation of the output. These three options work only on a PC with the Microsoft Internet Explorer browser currently. The **print_zoom**= option scales printed output. The **pagebreak**= option enables you remove the visible horizontal rule lines and the physical printed page breaks in the output.



The Date is Thursday, December 22, 2005

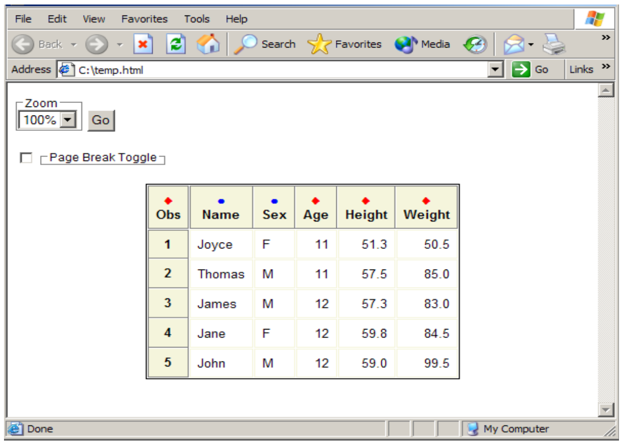
Obs	Name	Sex	Age	Height	Weight
1	Joyce	F	11	51.3	50.5
2	Thomas	M	11	57.5	85.0
3	James	M	12	57.3	83.0
4	Jane	F	12	59.8	84.5
5	John	M	12	59.0	99.5

```
options(print_header="The Date is &D" print_footer="The end" orientation="landscape" pagebreak="no"
print_zoom="40%")
```

Figure 8. Print Preview Window Showing Changed Output

CHANGING DISPLAY OPTIONS

The **describe=** option displays the data type of a column. The default is a blue circle for character and a red diamond for numeric. The **zoom=** option scales output on the screen, which is good if you want avoid a scroll bar. The **zoom_toggle**= option displays a selection list that enables the user to choose his or her own zoom property interactively. The **pagebreak_toggle=** option allows the page break to be toggled interactively. The **sort=** option enables you to click on the column headers to sort the data within the column. This option works only on a PC with the Microsoft Internet Explorer browser. The **design_mode=** option enables you to make the HTML editable. Finally, the **drag=** option allows tables and graphs to be moved around in the browser.


File Edit View Favorites Tools Help
Back Search Favorites Media
Address C:\temp.html Go Links
Zoom
100%
Go
Page Break Toggle
Done
My Computer

Obs	Name	Sex	Age	Height	Weight
1	Joyce	F	11	51.3	50.5
2	Thomas	M	11	57.5	85.0
3	James	M	12	57.3	83.0
4	Jane	F	12	59.8	84.5
5	John	M	12	59.0	99.5

```
options(describe="yes" zoom="70%" zoom_toggle="yes" pagebreak_toogle="yes")
```

Figure 9. Output Showing Multiple Display Options

CREATING A TREE VIEW OF THE TABLE OF CONTENTS

A tree view of the table of contents enables you to click on a plus or minus sign to navigate through the contents. By default, all table of contents entries are collapsed. The default image beside each entry is a plus or minus sign; however, you can use a different image by using the **open_image_folder=** option for the open image, the **closed_image_folder=** option for the closed image, and the **leaf_image_folder=** for the sub-entries. The **toc_background=** option modifies the background color of the table of contents. In the following PROC TEMPLATE code, the **htmlclass** attribute defines the tree view.

```
proc template;
 define style styles.tree;
  parent=styles.default;
   style contentprocname from contentprocname /
    htmlclass="expandable";
   style contentfolder from contentfolder /
    htmlclass="expandable";
   style bycontentfolder from bycontentfolder /
    htmlclass="expandable";
   style contentitem from contentitem /
    htmlclass="leaf";
  end;
 run;

ods tagsets.tableEditor file="c:\temp.html"
  contents="c:\temp1.html" style=styles.tree
options(open_image_path="file://c:\\closedfolder.gif"
        closed_image_path="file://c:\\openfolder.gif"
        toc_background="white") ;

proc print data=sashelp.class;
run;
proc print data=sashelp.class;
by age;
run;
proc univariate data=sashelp.class;
run;

ods _all_ close;
```

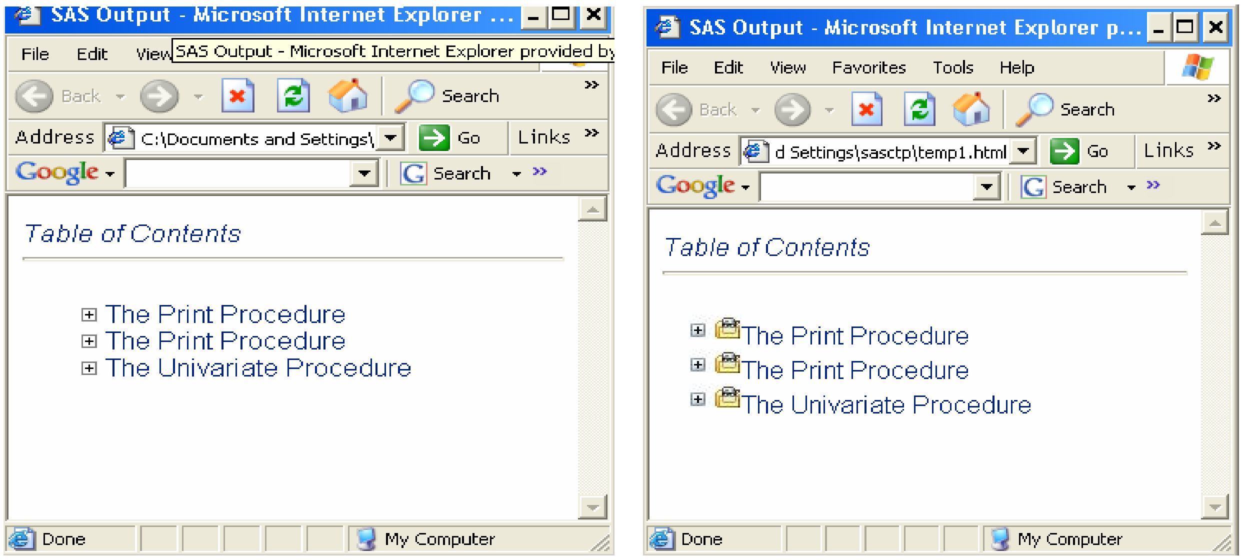


Figure 10. Output Showing a Tree View of the Table of Contents Using Default and Specified Images

TABLE EDITOR OPTIONS

The following is a partial list of options available in the Table Editor. You can find a complete list at ftp.sas.com/techsup/download/public/base/tableEditor.ZIP.

background_color=('string")
  specifies the background color of the body file.
background_image=("path")
  sets the image supplied as the background image.
banner_color_even=("string")
  adds background color for even rows.
banner_color_odd=("string")
  adds background color for odd rows.
closed_image_folder=("path")
  specifies the image to display when the table of contents entry is closed.
describe=(yes|no)
  displays the data type of the column. A blue circle is used for character and a red diamond for numeric.
design_mode=("yes|no")
  enables you to make the HTML editable.
drag=("string")
  allows tables and graphs to be moved within the browser.
frozen_headers=("yes|no")
  freeze column headers of the table. Is used with **pageheight=** option.
frozen_rowheaders=("yes|no")
  freeze row headers. Is used with **pagewidth=** option. This should be used only for a minimal amount of data.
highlight_color=("color")
  allows you to select a specific row or column and highlight it.
image_path=("image path")
  includes an image. The image is left-justified and added to the beginning of the document by default.
image_just=("left|center|right")
  justifies the image specified with the **image_path=** option.
leaf_image_folder=("path")
  specifies the image to display when the table of contents entry is a sub-entry.
open_image_folder=("path")
  specifies the image to display when the table of contents entry is open.

orientation=(“landscape|portrait”)
specifies the orientation of the output.
pagebreak=(“yes|no”)
enables you remove the visible horizontal rule lines and the physical printed page breaks in the output.
pagebreak_toggle=(“yes|no”)
toggles the virtual and physical page breaks in the document.
pageheight=(“height”)
determines the height of the page and how much data to display on the page when the page is scrolled.
pagewidth=(“width”)
determines the width of the page and how much data to display on the page when the page is scrolled.
print_zoom=(“scale”)
scales printed output.
print_header=(“string”)
specify text to be printed in the header of the printed output.
print_footer=(“string”)
specify text to be printed in the footer of the printed output.
scrollbar_color=("string")
modifies the scrollbar color from the default gray.
sort=(yes|no)
enables you to click on the column headers to sort the data within the column.
toc_color=("string")
modifies the background color of the table of contents.
zoom=(“scale”)
scales output on the screen.
zoom_toggle=(“yes|no”)
displays a selection list that enables the user to choose his or her own zoom property interactively.

WORKING WITH OTHER APPLICATIONS

CREATING MICROSOFT EXCEL FILES

The best way to create Microsoft Excel files from Base SAS software is to use the tagsets.ExcelXP tagset. This tagset generates output in the XML Spreadsheet format, which allows you to modify most of the options that can be selected from Microsoft Excel, but does not support graphics. To use tagsets.ExcelXP, Microsoft Excel 2002 or later and SAS 9.1 or later are required. This tagset allows you to create multiple worksheets per workbook that are not file-based. Each output object that is created generates a separate worksheet.

The MSOffice2k tagset can be used to generate HTML for Microsoft Excel 2000 and later and Microsoft Word 2000 and later. This tagset supports graphics, unlike the tagsets.ExcelXP tagset.

```
ods tagsets.ExcelXP file="temp.xls";
ods MSOffice2k file="temp1.xls";

proc print data=sashelp.class;
run;

proc print data=sashelp.air;
run;

ods _all_ close;
```

USING AUTOMATION TO CREATE MICROSOFT WORD AND MICROSOFT EXCEL FILES

Using automation, you can control applications directly from the script or tagset. For example, you can open Microsoft Excel and change formatting, rename a worksheet, add a chart, or change orientation. Automation is usually implemented with Visual Basic, but it can be implemented with scripting through VBScript and JScript.

In Visual Basic and VBScript, the **CreateObject()** function is used to enable automation. With JScript, the A**ctivexObject()** function is used to enable automation. In the next example, the **objectvar** variable is created and holds the instantiated object. C**lass** refers to the object, such as the Microsoft Excel application (excel.application).

```
   var objectvar=new ActivexObject(class);
```

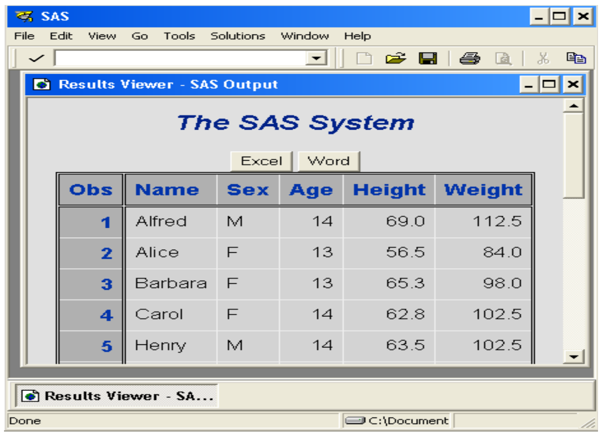
Everything that you can do in Microsoft Excel can be done using a script. The script can be included from an external file, or brought in-line using the DATA step or a tagset. Using a tagset allows more flexibility.

The following script uses the **createTextRange()** argument to include the entire body of the document. It uses **execCommand** to copy the body of the document to the clipboard. It then opens Microsoft Excel. Remember, any VBScript or JScript command from the object model can be used to modify the application. For example, in Microsoft Excel, you could freeze headers, modify the page setup, or save the file in its native application's format. In addition, you can pass parameters from the tagset as an argument to the script.

```
   // set a variable to hold the text range of the document */
   var tr = document.body.createTextRange();
   // select the text range
   tr.select();
   // copy to paste buffer
   document.execCommand("copy");
   // Instantiate a new instance of the Microsoft Excel application
   var xl = new ActiveXObject("excel.application");
   //  make the excel application visible
   xl.Visible = true;
   // Add a new workbook
   var wb = xl.Workbooks.Add();
   // Define active worksheet in workbook
   var sheet = wb.ActiveSheet;
   // Define where to start writing data
   sheet.Cells(1,1).Activate();
   //Name sheet
   sheet.Name="Wow";
   // Save the sheet.
   //sheet.SaveAs("C:\\TEST.XLS");
   //  Paste data from clipboard to active worksheet
   sheet.Paste();
```

In the next example, files are saved to Microsoft Excel and Microsoft Word.

```
    proc template;
      define tagset tagsets.print;
        parent=tagsets.msoffice2k;
         define event doc_head;
          start:
           put "<head>" NL;
           put VALUE NL;
           put "<script language=""javascript"">" NL;
           put "function CopyExcel()" NL;
           put "{" NL;
           put " var tr = document.body.createTextRange();" NL;
           put " tr.select();" NL;
           put " document.execCommand(""copy"");" NL;
           put " var xl = new ActiveXObject(""excel.application"");" NL;
           put " xl.Visible = true;" NL;
           put " var wb = xl.Workbooks.Add();" NL;
           put " var sheet = wb.ActiveSheet;" NL;
           put " sheet.Cells(1,1).Activate();" NL;
           put " sheet.Paste();" NL;
           putq " sheet.Name=" $options['SHEET_NAME'] ";" NL;
           putq " sheet.SaveAs("  $options['SAVEAS'] ");"  NL;
           put "}" NL;

           put "function CopyWord()" NL;
           put "{" NL;
           put "var tr = document.body.createTextRange();" NL;
           put "tr.select();" NL;
           put "document.execCommand(""copy"");" NL;
           put "var word = new ActiveXObject(""word.application"");" NL;
           put " word.Visible = true;" NL;
           put " word.Documents.Add();" NL;
           put " word.Selection.Paste();" NL;
           put " word.Quit();" NL;
           put "}" NL;
           put "</script>" NL;
         finish:
           put "</head>" NL;
          end;
         end;
         define style styles.test;
           parent=styles.default;
             style table from table /
               prehtml='<input type=button value="Excel" onclick="CopyExcel()">
                        <input type=button value="Word"  onclick="CopyWord()"> ';
            end;
        run;

       ods tagsets.print file="temp.html" style=styles.test
       options(saveas="c:\\temp.xls"
               sheet_name="wow");

       proc print data=sashelp.class;
       run;

       ods tagsets.print close;
```



The SAS System

Obs	Name	Sex	Age	Height	Weight
1	Alfred	M	14	69.0	112.5
2	Alice	F	13	56.5	84.0
3	Barbara	F	13	65.3	98.0
4	Carol	F	14	62.8	102.5
5	Henry	M	14	63.5	102.5

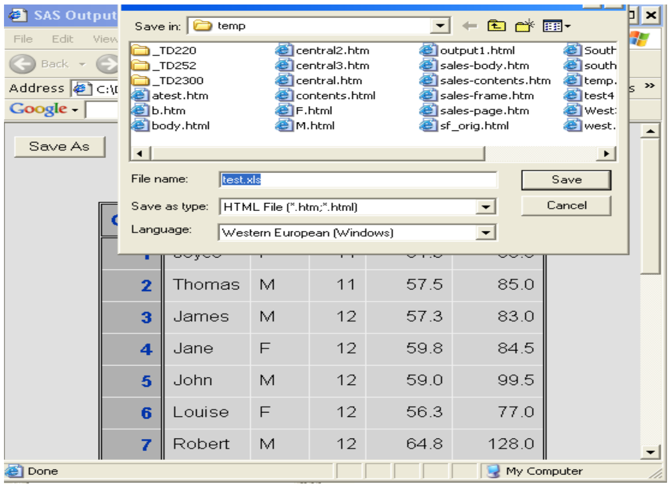
SAS
File Edit View Go Tools Solutions Window Help
Results Viewer - SAS Output
Excel
Word
Results Viewer - SA...
Done
C:\Document

Figure 11. Output Showing the Capability to Save to Microsoft Excel and Microsoft Word

SAVING MICROSOFT EXCEL FILES FROM A WEB PAGE

HTML files can be saved as Microsoft Excel files with the **document.execCommand** function and the SAVEAS argument. This function opens a Save AS dialog box, which includes the name of the file to download. To save the file as a Microsoft Excel file, add the extension of **.xls** or **.csv** and a Microsoft Excel version of the HTML file is saved (not a native Excel file). If you want to name the file before the dialog box is opened, you can specify the filename and extension as an argument in **document.execCommand**. Any single slashes in the path need to be replaced with double slashes.

```
/* Syntax which names file */

proc template;
 define style styles.test;
  parent=styles.default;
   style body from body /
    prehtml='<input onclick="document.execCommand(''saveas'',true,
              ''c:\\temp\\test.xls'')" value="Save As" type="button">';
  end;
run;

ods msoffice2k file='temp.html' style=styles.test;
proc print data=sashelp.class;
run;
ods msoffice2k close;
```



File name: test.xls
Save as type: HTML File (*.htm;*.html)
Language: Western European (Windows)

2	Thomas	M	11	57.5	85.0
3	James	M	12	57.3	83.0
4	Jane	F	12	59.8	84.5
5	John	M	12	59.0	99.5
6	Louise	F	12	56.3	77.0
7	Robert	M	12	64.8	128.0

Figure 12. The Save in Dialog Box

XML IN STYLE

XML has been used extensively for transferring data. XML can be read and written from SAS using the XML Libname Engine, and written using the ODS MARKUP destination. When you view XML output, what you see is a file full of tags. To present the data, you need to apply style to the output; there are two methods--through CSS or XSL.

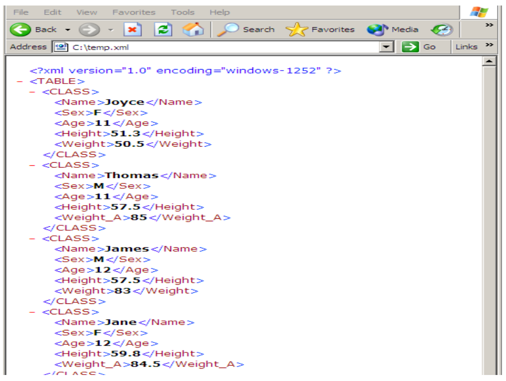
The next example uses CSS. The output is created with the XML engine and the libname statement. The **Name** is bolded. Values are placed on separate lines by using **display:block** CSS style property and value. Values are emphasized by using **font-size:larger** and **font-weight:bold** style properties. A box has been added around each observation by using the **border** property, and spacing is added using the **margin**, and **padding** properties for the **Class** node . A universal style selector ***** has been added to apply a font to all of the output.

```
Class       {display:block;
             border:2px solid black;
             margin:15px;
             padding:7px;
             padding-bottom:40px;
             width:200px;}
```

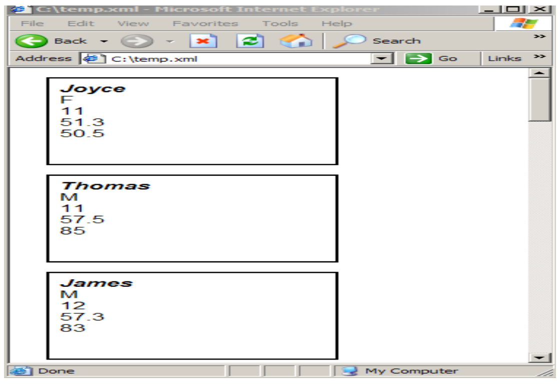
```
Name        {display: block;
             font-weight:bold;
             font-style:italic;}

Sex         {display:block;}
Age         {display:block;}
Height      {display:block;}
Weight      {display: block;}

*  {font-family:Arial, Helvetica, sans-serif;}
```

In the next example, PROC TEMPLATE applies CSS to the XML file.

```
proc template;
 define tagset tagsets.testcss;
  notes "sas xml model";
   define event xmlversion;
    put "<?xml version=""1.0""";
    putq "encoding=" ENCODING;
    put " ?>" NL;
    put "<?xml-stylesheet type='text/css' href='class.css'?>" NL;
    break;
   end;
   parent=tagsets.sasxmog;
   end;
run;

libname temp xml 'c:\temp.xml' tagset=tagsets.testcss;

data temp.class;
 set sashelp.class;
run;
```



```
  <?xml version="1.0" encoding="windows-1252" ?>
- <TABLE>
 - <CLASS>
     <Name>Joyce</Name>
     <Sex>F</Sex>
     <Age>11</Age>
     <Height>51.3</Height>
     <Weight>50.5</Weight>
   </CLASS>
 - <CLASS>
     <Name>Thomas</Name>
     <Sex>M</Sex>
     <Age>11</Age>
     <Height>57.5</Height>
     <Weight_A>85</Weight_A>
   </CLASS>
 - <CLASS>
     <Name>James</Name>
     <Sex>M</Sex>
     <Age>12</Age>
     <Height>57.5</Height>
     <Weight>83</Weight>
   </CLASS>
 - <CLASS>
     <Name>Jane</Name>
     <Sex>F</Sex>
     <Age>12</Age>
     <Height>59.8</Height>
     <Weight_A>84.5</Weight_A>
```


C:\temp.xml - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back
Search
Address C:\temp.xml
Go
Links
Done
My Computer

```
Joyce
F
11
51.3
50.5

Thomas
M
11
57.5
85

James
M
12
57.3
83
```

Figure 13. XML Output and XML Output with CSS Style Applied

XSL STYLE SHEETS

XSL stands for Extensible Stylesheet Language. XSL enables you to change the structure of how a file is presented, such as displaying an XML file in an HTML or other format. The XSL format makes use of the XPATH syntax to locate the data to present. XLS allows great flexibility, but is not limited to filtering, sorting, searching, and general styling of output. This allows us to create a template which describes how our data should be presented. The functionality of XSL to XML is similar to the functionality that CSS has to HTML, however, XSL can be used to totally transform this file. The other half of XSL includes XSL Formatting Objects (XSL-FO) which allows the creation of formatting objects that gives complete control over the layout of a page. This format can be imported in PDF.

There are several ways that XSL files can be applied to an XML file to change how the data is presented. You can use the ODS Markup. Several transformation engines enable you to create a single file from the XSL file and XML file. Scripting and various options can be used.

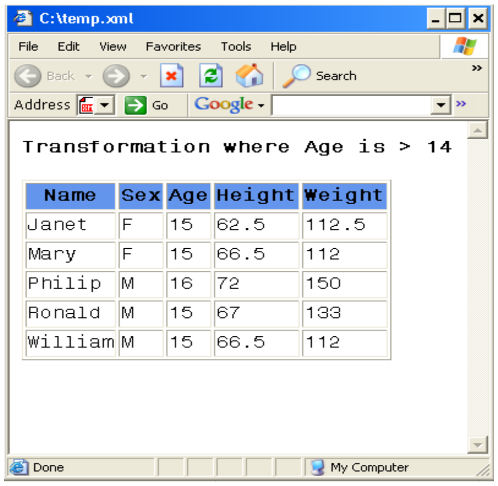
The next example shows how an XSL style sheet is used to transform an XML file, which was created from the **sashelp.class** data set, to an HTML file. The **for-each select** code performs a loop through each node and selects the value from each node with the **value-of select** statement. The value of the **test** attribute in the conditional **if** expression outputs the value of **age** when it is greater than 14.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
 <html>
 <body>
  <h4>Transformation where Age is > 14</h4>
  <table border="1">
   <thead>
    <tr bgcolor="#6495ED">
     <th>Name</th>
     <th>Sex</th>
     <th>Age</th>
     <th>Height</th>
     <th>Weight</th>
    </tr>
   </thead>
 <xsl:for-each select="TABLE/CLASS">
 <xsl:if test="Age&gt;14">
  <tbody>
   <tr>
    <td><xsl:value-of select="Name"/></td>
    <td><xsl:value-of select="Sex"/></td>
    <td><xsl:value-of select="Age"/></td>
    <td><xsl:value-of select="Height"/></td>
    <td><xsl:value-of select="Weight"/></td>
   </tr>
  </tbody>
 </xsl:if>
 </xsl:for-each>
   </table>
 </body>
 </html>
</xsl:template>
</xsl:stylesheet>
```

In the next example, PROC TEMPLATE applies XSL to the XML file.

```
ods path(prepend) work.templat(update);
proc template;
 define tagset tagsets.addxsl;
  notes "SAS XML model";
   define event XMLversion;
    put "<?xml version=""1.0""";
    putq "encoding=" ENCODING;
    put "?>" NL;
    put "<?xml-stylesheet type='text/xsl' href='class.xsl'?>" NL;
   break;
  end;
  parent=tagsets.sasxmog;
  end;
 run;
```

```
libname temp xml 'c:\temp.xml' tagset=tagsets.addxsl;

data temp.class;
 set sashelp.class;
run;
```



Transformation where Age is > 14

Name	Sex	Age	Height	Weight
Janet	F	15	62.5	112.5
Mary	F	15	66.5	112
Philip	M	16	72	150
Ronald	M	15	67	133
William	M	15	66.5	112

Figure 14. Output Showing XSL Applied to an XML File

E-MAILING FILES

E-mailing HTML files can be done using the **email** access device of the **filename** statement to send HTML as the body of the e-mail. You can also send an HTML file as an attachment. To display the HTML as the body of the e-mail, use the options **emailsys** and **emailhost** with the **type=** option. The **emailsys** option should be set to the SMTP and the **emailhost** option should specify the name of the mail host.

```
filename temp email to="joebloe@sas.com"
                    type="text/html"
                    subject="testing";
ods html file=temp;

proc print data=sashelp.class;
run;
ods html close;
```

CONCLUSION

This document discusses a variety of ways to add style to your output. Whether you want to enhance the printed output, the viewed output, XML output, or another application's output, there are several options available for you to use. Hopefully, you have found something in this document that will allow you to say "Now–That's My Style."

REFERENCE

"*ScriptX Printing*". October 2005. MEDCO INC." Available www.meadroid.com/scriptx/docs/printdoc.htm.

RECOMMENDED READING

"*How To Automate Excel from an HTML Web Page Using JScript*". August 2005. Microsoft. Available support.microsoft.com/default.aspx?scid=kb;en-us;234774.

"ODS FAQ's and Concepts." SAS Institute Inc. Available support.sas.com/rnd/base/topics/templateFAQ/Template.html.

"ODS MARKUP Resources." SAS Institute Inc. Available support.sas.com/rnd/base/topics/odsmarkup/.

"XML." SAS Institute Inc. Available support.sas.com/rnd/base/index-xml-resources.html.

CONTACT INFORMATION

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

Chevell Parker
SAS Institute Inc.
SAS Campus Drive
Cary, NC 27513
Chevell.Parker@sas.com