Paper 164-26

Avoiding a (Graphic) Identity Crisis with ODS HTML Styles

Jaclyn Whitehorn, The University of Alabama, Tuscaloosa, Alabama

ABSTRACT

The new SAS® Output Delivery System (ODS) enables any SAS programmer to produce professional-quality output. In particular, the ability to generate HTML output directly from every SAS procedure was highly anticipated by the SAS user community. However, it is doubtful that the default HTML output fulfills any site's Web publication standards. The capability of defining a custom ODS style is provided through PROC TEMPLATE, but it is not for the faint-of-heart. There are a number of new concepts to learn, and the amount of documentation is small (but growing). Furthermore, there are some odd inheritance structures in the default templates provided by SAS Institute, which makes editing them a greater challenge than necessary. This paper is intended to serve as a "getting started" guide for those users that need to modify the default templates in order to create their own styles. It also introduces an alternative style inheritance structure.

INTRODUCTION

You're so proud of yourself! You have created your first HTML output using the new SAS Output Delivery System. You happily envision your new future without 200-page printouts, HTML macros, or cut-and-paste. But your balloon is quickly deflated when the verdict comes down from your boss: "Well, it's kind of *ugly*. Can't you do something about that?"

Face it: an attractive Web page design generally consists of more than just various shades of gray. And even if your entire department is seriously artistically deprived and doesn't mind the gray, the institution you work for probably has pre-existing Web graphic standards. (And if it doesn't yet, it probably will soon!) So now your task is to take your new capability of producing HTML output and use it to create *attractive* HTML output (or at least output that conforms to your department's standards).

If you don't know where to start, or if you took one look at the PROC TEMPLATE documentation and the default style definition and threw up your hands in despair, this paper is for you. First, we will learn about the parts of a template. Then we will discuss the steps involved in creating your own style. Finally, we will consider ways to make the whole process easier by producing new default templates as starting points. While it cannot make you a full-fledged Web designer, hopefully this will put you on the right track for creating attractive, consistent HTML output.

REQUIRED PRIOR KNOWLEDGE

You should have a working knowledge of Base SAS and be able to produce some output that you would like to publish in HTML format.

While it is not strictly necessary, a good grounding in HTML will save you a lot of headaches later on. It is very difficult to tell the Output Delivery System how to write HTML if you do not know how to do it yourself. Even simple things like changing a font or color will be difficult unless you know how to create a Web page outside of the SAS System. Some experience designing Web pages will also help you make your styles more attractive and user-friendly, a skill that can only be learned by observing other Web sites and practicing making your own. If you think you will be doing extensive modifications, you should also learn about Cascading Style Sheets (CSS). There are many excellent HTML/CSS tutorials and references available both online and in hardcopy.

Finally, this paper will not discuss publishing HTML output to a Web server. This is mainly due to the wide variety of Web server configurations in use. Your institution's computer network support should be able to assist you in publishing your results. If you are doing this privately, you will need an account with an Internet Service Provider (ISP) that includes space for a Web site and technical assistance for doing so.

CREATING ODS HTML OUTPUT

Before you can attempt to customize ODS output, you have to know how to create it. The following code fragment closes the LISTING destination, opens the HTML destination, and specifies where to send the output.

```
ods listing close;
ods html body='body.html' contents='toc.html'
 page='page.html'
 frame='frame.html'(title='My SAS Output')
 path='directory to put files in'(url=none)
 style=Beige;
```

Closing the LISTING destination means that no output will go to the output window or listing file. The four files named in the BODY=, CONTENTS=, PAGE=, and FRAME= options will be created in the directory given in the PATH= option. (If files of those names already exist, they will be overwritten.) The value of the TITLE= suboption for the frame file will appear in the browser title area and be used as the name for any bookmarks or "favorites" pointing to it, but it will not print on the page. The URL=NONE suboption is necessary to ensure that links between the four files do not use absolute references, which include the directory path, so that you can publish the HTML output to a Web server. The STYLE= option specifies which style template to use; this example uses the "Beige" template supplied by SAS Institute. If the STYLE= option is omitted, the output will be generated using the default style.

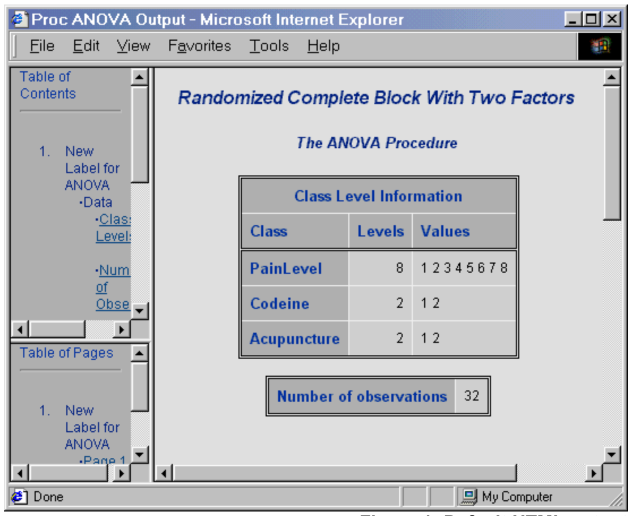
For those that are unfamiliar with HTML frames, some explanation may be useful. Each of the four files that the ODS creates (body.html, toc.html, page.html, and frame.html) is an HTML file. The body file is where all of the output will go. In fact, if you do not want a framed page, this is the only file that you need to request. The contents file is a table of contents to each piece of output. The page file is an index to the separate "pages" of the output file. These pages correspond to the separate pages that would have been created in normal output, not to where the HTML file will break when printed. You may have a table of contents, or a table of pages, or both, or neither. The frame file has no content of its own and is only needed if you have a contents or page file. It acts to hold all of requested pages in one browser window. To see your output, point your Web browser to the frame file. The default output style places the body file on the right and the table of contents and page index on the left (see Figure 1).

For more information about using the Output Delivery System and/or the HTML destination, please see SAS Institute's *The Complete Guide to the SAS Output Delivery System*.

TEMPLATES AND TEMPLATE STORES

There are two types of templates utilized by the Output Delivery System: table templates and style templates. Table templates control the display of output created by individual procedures and data steps. Style templates, on the other hand, control aspects of the overall presentation, such as background colors and fonts. At this point we are only interested in creating a visual identity with style templates.

Templates are saved in a template store, which is a special type of SAS file. An ODS PATH statement specifies which template stores are in use by that SAS session. In general, whenever a style is named for reading, the ODS searches each template store in the PATH, in order, until it finds one with that name. It uses the first one it finds and stops searching. When you need to store a style, the ODS will put it in the first template store in the PATH to which you have the proper access. (“Write” access will only write new styles; “update” access is required to make changes to styles that already exist in that store.)



Randomized Complete Block With Two Factors

The ANOVA Procedure

Class Level Information

Class	Levels	Values
PainLevel	8	1 2 3 4 5 6 7 8
Codeine	2	1 2
Acupuncture	2	1 2

Number of observations	32

Figure 1: Default HTML output

If there will be more than one computer needing access to your new style, you should put it in a template store in an accessible place like a shared network drive. Then you will need to add the new template store to the ODS PATH statement for each computer that needs to use it to produce output; depending on your local configuration, this may be most easily accomplished with an autoexec.sas file.

The following code assumes that a template store exists in the library named on the first line. (We will be creating that template store later.) It gets added to the PATH with read access, which would be appropriate for general use. By listing it first, we tell ODS to search the new template store before the others. This means that output run without a style specified would look for a style named styles.default in the template store on the network drive before using the one supplied by SAS Institute. (All pre-installed templates are stored in sashelp.tmplmst.)

```
libname ourstyle 'network directory';
ods path ourstyle.templat(read)
 sasuser.templat(update)
 sashelp.tmplmst(read);
```

HOW STYLE TEMPLATES WORK

A style template is primarily made up of style elements, each with attributes. Many elements serve to render a particular part of the HTML output, but some exist simply to provide a starting point for other elements. (SAS Institute calls these “abstract” elements.) For example, the “Cell” element in the default style is abstract, so it does not directly affect any visible output. However, the element “Data” controls the rendering of table cells containing data, and it is based on the “Cell” element. Other style elements that control special data cells are then based on the “Data” element. This is the first of two types of style *inheritance*; the second will be discussed shortly.

Style attributes control different aspects of each style element. Different attributes specify a font for the element (“font”), the font color (“foreground”), and a background color (“background”), to name a few. When one style attribute is based on another, it inherits all the attributes from the first one, but changes and additions can still be made.

Style templates are defined in a PROC TEMPLATE step. A STYLE statement is used in a PROC TEMPLATE step to define a style element. In the following example, the “Cell” element specifies a background and foreground color. The “Data” element inherits both of those, but then changes the foreground color. A data cell in output using this style would have a white background and gray text. (This code will not run by itself; it is part of a PROC TEMPLATE step.)

```
style Cell /
 background = white
 foreground = black ;
style Data from Cell /
 foreground = gray ;
```

Another type of style element used is a reference list. These are similar to associative arrays or “hashes” found in other languages like Perl. You can think of reference lists as providing nicknames for more complicated structures. One of the reference lists in the default style is called “fonts.” This list sets up names like “docfont” and “TitleFont” to refer to particular font faces and sizes. In the following code fragment, “docFont” is set up to refer to font faces Arial, Helvetica, or Helv in HTML size 3, while the “TitleFont” is Arial, Helvetica, or Helv in HTML size 5, bolded and italicized. Then the “Cell” element has its font attribute set to “docFont,” which is then inherited by the “Data” element.

```
style fonts /
 'docFont' = ("Arial, Helvetica, Helv", 3)
 'TitleFont' = ("Arial, Helvetica, Helv",
               5, Bold Italic) ;
style Cell /
 background = white
 foreground = black
 font = fonts('docFont');
style Data from Cell /
 foreground = gray ;
```

Reference lists are also used in the default style to define shortcuts to colors and to commonly used text and HTML.

For the second type of style inheritance, a style definition begins by declaring a “parent” style. The style being defined is the “child” of the parent style and inherits all of its elements. New elements can still be defined, and existing elements can be changed within the child style. The first way of doing this is to use the STYLE statement like before. The STYLE statement only changes the element named, *not any that may inherit from it.* Usually you will base an element in the child style on the element of the same name from the parent style. This allows you to only include the attributes that need to be added or changed. The following complete PROC TEMPLATE step bases a new style on the default and changes the font in the data cells.

```
proc template;
define style styles.new;
 parent = styles.default;
 style Data from Data /
  font = ("Times New Roman, Times, serif",3);
end;
run;
```

If you want other elements to inherit attributes that you set in the child style, you need to use the REPLACE statement instead of the STYLE statement. When you replace an element, the style template is built as if the element in the child template completely replaces the element of the same name from the parent template. Because of this, any attribute set for that element in the parent style will need to be reset in the element that is replacing it, even if it otherwise would not be changed. However, you can still base the new element definition on the one of the same name from the parent style in order to preserve the inheritance structure. The following example replaces the background color to the element “Data,” ensuring that any element that inherits from it gets the new background color. Note that this example uses a hexadecimal code to represent the color. This is used much more frequently than a color name.

```
proc template;
define style styles.new;
 parent = styles.default;
 replace Data from Data /
  foreground = colors('datafg')
  background = cxFFCCCC; /* light pink */
end;
run;
```

READING THE DEFAULT STYLE

We now have most of the information needed to start creating a style of our own. However, most people will not make one from scratch. There are a large number of elements, many of which you may never work with, that are needed to make a style template compile without warnings or notes in the log. To avoid these problems, it is best to base your style off of one that already has all the elements in it, like the default template provided by SAS Institute. You need to know how the template is defined before trying to edit it, so use the following code to save the source for the default template:

```
proc template;
source styles.default /
 file='external file location';
run;
```

You will see that the source code is quite long and can get very confusing. All the same, it is important to read through it to get some idea of the inheritance structure. Some of the more important style elements are listed in Table 1.

Some of the elements, like “SystemTitle” and “ProcTitle,” have associated “containers” as well. The Output Delivery System places each of these elements inside its own one-cell table, and each table is rendered by a container element that describes its width and border. The background color is controlled by the main element, not the container. The “SysTitleandFooterContainer” element holds the “SystemTitle,” while the container for “ProcTitle” is the “TitleandNoteContainer.” Figure 2 (at the end of the text) shows the position of many style elements in relation to the rest of the output.

More details about the default style template supplied by SAS Institute can be found under PROC TEMPLATE: Concepts in *The Complete Guide to the SAS Output Delivery System*.

BEGINNING YOUR OWN STYLE

Assuming that you have already submitted the two statements under “About SAS Templates” to assign a library and set the ODS path, start your PROC TEMPLATE with the following:

```
libname ourstyle 'network directory';
proc template;
path sashelp.tmplmst(read);
define style styles.test /
 store=ourstyle.templat;
 parent = styles.default;
```

The PATH statement temporarily changes the ODS PATH used; this ensures that the “styles.default” that the style is based on is the one supplied by SAS Institute. The STORE statement saves the styles.test template in the shared template store.

In order for other people to be able to help you test this style, you must first have set up their ODS PATH statements as detailed above. Then, they must use the STYLE=styles.test option in their ODS HTML statement. Once you have your style completed, tested, and approved (if necessary), you may want to name it “styles.default.” Since the ODS PATH statement names ourstyle.templat first, any time ODS is used to produce HTML output, it will find and use your default style instead of the one from SAS Institute. Therefore, the STYLE= option in the ODS HTML statement will no longer be necessary.

After the PARENT statement, include any STYLE or REPLACE statements you need to customize the output. The abbreviated list of style element attributes in Table 2 may be useful to you. Keep in mind that not all attributes make sense with all elements. “Mismatched” attributes will not give errors; they will just not have any effect.

ALTERNATE INHERITANCE STRUCTURES

Some people have found the inheritance structure built into the templates provided by SAS Institute to be non-intuitive. While some of the complexity stems from the inherit flexibility of ODS output, alternate style templates can be created to better facilitate customization.

A major problem with the current structure is that most elements do not inherit defaults from the file in which they are contained. If you set a font or foreground color for the “Body” element, no other element inherits it. While the solution is readily apparent from studying the source code for the default template, this type of problem creates a steeper learning curve and may discourage some users from attempting to create their own styles.

Another issue is the use of two reference lists for colors. While the two-level redundancy may be more efficient for higher-level programmers, it adds unneeded complexity to the process for many users. Multiple pieces of output that may look related and be the same color by default can actually refer to two different entries in the “colors” reference list, thus breaking some of the benefits of inheritance.

In an attempt to make it easier for more users to create their own templates, the author has produced two templates featuring an alternate inheritance structure. One is a stripped-down version of the SAS default; most users will not see any cosmetic differences in output. The other has very little in the way of attributes and can be used as an inheritance framework. Those templates and their documentation can be found at <http://bama.ua.edu/~jaclyn/sasods>. The author encourages feedback on these templates and will attempt to keep them up-to-date with needed improvements.

CONCLUSION

Like the rest of The SAS System, the Output Delivery System is extremely powerful and flexible. Through PROC TEMPLATE, users have the ability to customize the overall style of HTML output so that it fits with their sites’ other Web material. This is not a simple process, but the result is worth it: a reusable style to quickly get any SAS output you or your team produces ready for publication on the World Wide Web.

There is plenty of room for innovation within the Output Delivery System. While the author hopes that her templates can be of some use, they should also initiate a discussion about other possible inheritance structures. There is also a need for additional style templates that could be used either as-is, or as starting points for additional modification. As of January 2001, there were no user-developed styles available in SAS Institute’s ODS Style Gallery <http://www.sas.com/rnd/base/index-gallery.html>. The ability to create professional output directly from SAS has been anticipated for some time; now the very talented population of SAS programmers should expand on it.

REFERENCES

A comprehensive source of reference information on the Output Delivery System (including PROC TEMPLATE) is:

SAS Institute Inc., *The Complete Guide to the SAS Output Delivery System*, Cary, NC: SAS Institute, Inc., 1999

Other available sources:

The Template FAQ, http://www.sas.com/rnd/base/topics/templateFAQ/Template.html

Olinger, Christopher, “Twisty Turny Passages, All Alike – PROC TEMPLATE Exposed,” *Proceedings of the Twenty-Fourth Annual SAS Users Group International Conference*, 1999

The Base SAS Community Web site, http://www.sas.com/rnd/base/

The output used for Figure 1 is a basic ANOVA adapted from an example in the SAS/STAT® User’s Guide portion of SAS Institute Inc., *SAS OnlineDoc®, Version 8*, Cary, NC: SAS Institute Inc., 1999. It was modified so that it would only produce HTML output.

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are registered trademarks or trademarks of their respective companies.

CONTACT INFORMATION

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

Jaclyn Whitehorn
The University of Alabama
Box 870346
Tuscaloosa, AL 35487-0346

Work Phone: (205) 348-8720
Fax: (205) 348-3993

E-mail: jaclyn@bama.ua.edu
Web: http://bama.ua.edu/~jaclyn/sasods

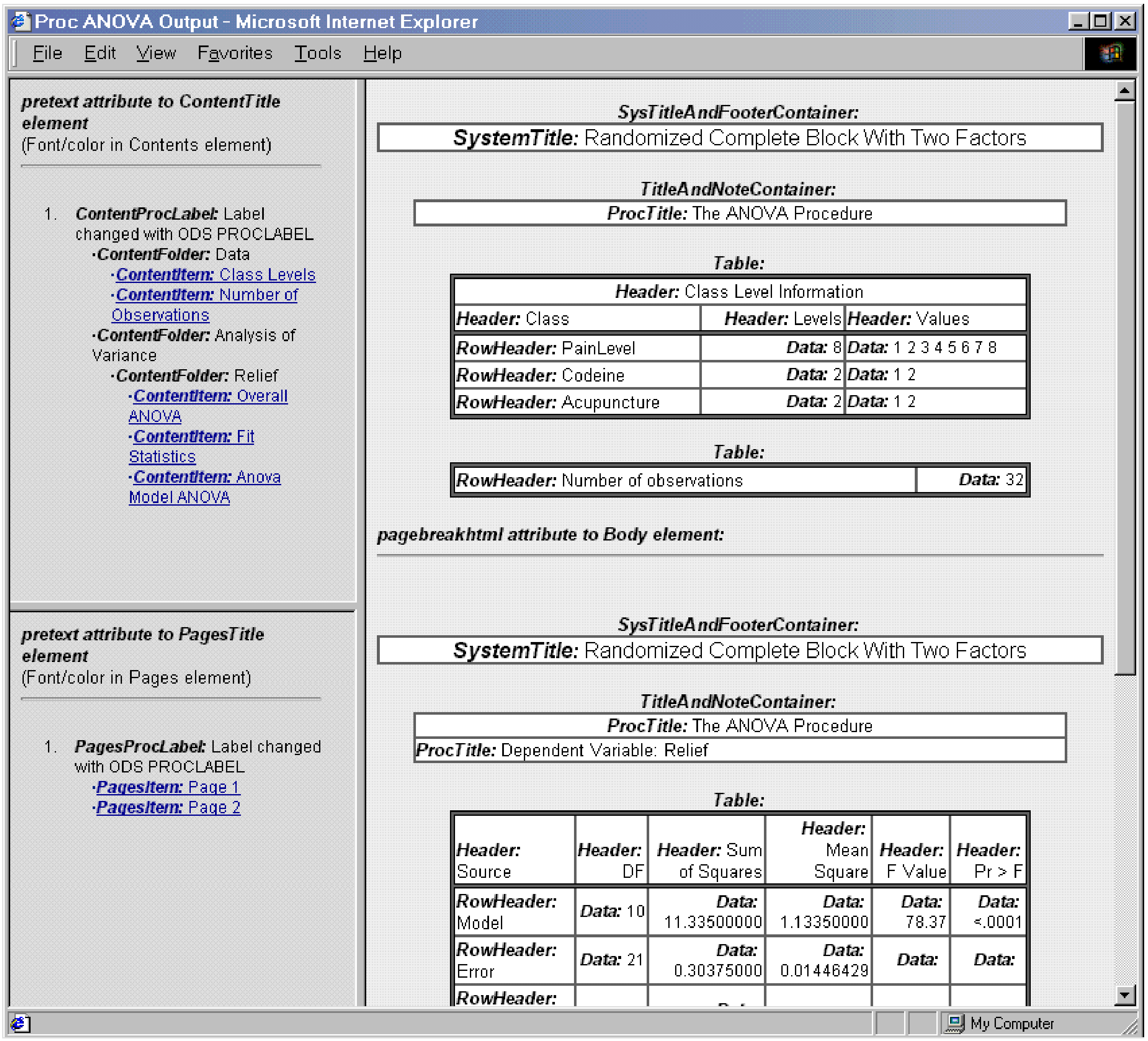
TABLES

TABLE 1: SELECTED STYLE ELEMENTS

Style Element	Description
Body, Frame, Contents, Pages	Control appearance of the various output files (background, default font, etc.)
ContentTitle, PagesTitle	Control the titles for the Table of Contents and Pages files.
SystemTitle	Controls appearance of system-provided titles (such as those produced by TITLE statements)
ProcTitle	Controls appearance of procedure-defined titles
Header, RowHeader	Renders table cells that act as column or row headers
Data	Renders table cells that contain data
ContentProcName, PagesProcName	Controls appearance of procedure names in Table of Contents and Pages files. (If you use the ODS PROCLABEL statement to change the text used to label the procedure, you will want to edit the ContentProcLabel and PagesProcLabel elements instead.)

TABLE 2: SELECTED ELEMENT ATTRIBUTES

Attribute	Description
font	Set the font for the element. Requires font list in the form (“font-face list”, font-size, keyword list). Font faces should be separated by commas. Sizes are given in HTML form, 1 through 7. Keywords pertain to font weight and style, such as bold or italic. A reference to the “font” list can also be used.
background	Background color. For tables, sets the color between cells.
foreground	Usually controls font color, border color for containers
frame	Outside border for tables/containers. Common settings are “VOID” (no border) or “BOX” (border all the way around).
linkcolor, visitedlinkcolor	Sets colors for unvisited and visited links. There is apparently no attribute for active links.
cellspacing, cellpadding	Sets the space between and within cells, respectively. The color of the cellspacing is set by the table background, while the padding is controlled by the cell background.
pretext, posttext, prehtml, posthtml	Inserts text or HTML code before or after the element. Note that the use of “before” or “after” can be a little deceptive. The text of the “ContentTitle” is contained in “pretext,” while you can use “prehtml” with “Body” to insert a custom page header.


Proc ANOVA Output - Microsoft Internet Explorer
File Edit View Favorites Tools Help
My Computer

pretext attribute to ContentTitle element
(Font/color in Contents element)

1. ***ContentProcLabel:*** Label changed with ODS PROCLABEL
   - ***ContentFolder:*** Data
     - ***ContentItem:*** Class Levels
     - ***ContentItem:*** Number of Observations
   - ***ContentFolder:*** Analysis of Variance
     - ***ContentFolder:*** Relief
       - ***ContentItem:*** Overall ANOVA
       - ***ContentItem:*** Fit Statistics
       - ***ContentItem:*** Anova Model ANOVA

pretext attribute to PagesTitle element
(Font/color in Pages element)

1. ***PagesProcLabel:*** Label changed with ODS PROCLABEL
   - ***PagesItem:*** Page 1
   - ***PagesItem:*** Page 2

SysTitleAndFooterContainer:

SystemTitle: Randomized Complete Block With Two Factors

TitleAndNoteContainer:

ProcTitle: The ANOVA Procedure

Table:

Header: Class Level Information		
Header: Class	***Header:*** Levels	***Header:*** Values
RowHeader: PainLevel	***Data:*** 8	***Data:*** 1 2 3 4 5 6 7 8
RowHeader: Codeine	***Data:*** 2	***Data:*** 1 2
RowHeader: Acupuncture	***Data:*** 2	***Data:*** 1 2

Table:

RowHeader: Number of observations	***Data:*** 32

pagebreakhtml attribute to Body element:

SysTitleAndFooterContainer:

SystemTitle: Randomized Complete Block With Two Factors

TitleAndNoteContainer:

ProcTitle: The ANOVA Procedure

ProcTitle: Dependent Variable: Relief

Table:

Header: Source	***Header:*** DF	***Header:*** Sum of Squares	***Header:*** Mean Square	***Header:*** F Value	***Header:*** Pr > F
RowHeader: Model	***Data:*** 10	***Data:*** 11.33500000	***Data:*** 1.13350000	***Data:*** 78.37	***Data:*** <.0001
RowHeader: Error	***Data:*** 21	***Data:*** 0.30375000	***Data:*** 0.01446429	***Data:***	***Data:***
RowHeader:					

Figure 2: HTML output labeled with attribute names