# Contents

*About This Book* .............................. vii

**PART 1  Advanced PROC DOCUMENT Topics**  1

*Chapter 1  •  Introduction* .......................................................... 3
  Advanced PROC DOCUMENT Features .................................................. 3

*Chapter 2  •  Restructuring Output with PROC DOCUMENT* ...................... 5
  Overview .......................................................................................... 5
  Viewing an ODS Document in the Documents Window ................................ 5
  Viewing the Contents of an ODS Document ............................................. 6
  Using the Documents Window Pop-up Menu .......................................... 7
  Examples ......................................................................................... 11

*Chapter 3  •  Using WHERE Expressions in PROC DOCUMENT* ................ 21
  Overview ......................................................................................... 21
  WHERE Expression Syntax .................................................................... 21
  Example: Opening and Listing ODS Documents Using WHERE Expressions .... 25

*Chapter 4  •  Working with Output Objects and Table Templates* ............... 29
  Overview ......................................................................................... 29
  Displaying the Template Used by Output Objects .................................... 29
  Example: Using PROC DOCUMENT to Work with Table Templates ............ 30

**PART 2  ODS and Cascading Style Sheets**  37

*Chapter 5  •  Introduction to Cascading Style Sheets* ............................. 39
  Overview ......................................................................................... 39
  Terminology ...................................................................................... 41

*Chapter 6  •  Working with the ODS Document Object Model* .................. 45
  ODS Document Object Model (DOM) Structure ....................................... 45
  Tips and Things to Remember ........................................................... 47

*Chapter 7  •  Using Style Selectors* ..................................................... 49
  Overview ......................................................................................... 49
  Class Selectors .................................................................................. 50
  Element Selectors ................................................................................ 50
  Universal Selector .............................................................................. 51
  Pseudo-Class Selectors ....................................................................... 51
  ID Selectors ...................................................................................... 52
  Attribute Selectors ............................................................................. 52
  Combinators ...................................................................................... 53
  Using a Single Style Selector Type per Rule Set ....................................... 54
  Using Multiple Style Selector Types per Rule Set .................................... 55
Contents

Tips for Working with Style Selectors ........................................ 55

Chapter 8 • SAS Dynamic Property Values .................................. 57
    SAS Dynamic Property Values ............................................ 57

Chapter 9 • Resources and Examples ......................................... 61
    Common Industry-Standard Style Properties .......................... 61
    Documentation Resources and Websites ................................. 73
    Examples ................................................................................ 73

PART 3  Report Writing Interface 109

Chapter 10 • Introduction to the Report Writing Interface ............... 111
    Overview ............................................................................. 111
    Terminology ........................................................................ 112
    Types of Methods .................................................................. 113
    ODS Output Object Creation .............................................. 118

Chapter 11 • Reference and Examples ......................................... 121
    ODS Output Object Creation .............................................. 122
    Types of Methods .................................................................. 123
    Statements and Methods by Category .................................... 128
    Dictionary ............................................................................. 130
    Examples .............................................................................. 195

Chapter 12 • Concepts .............................................................. 249
    RWI Basics .......................................................................... 249
    DATA Step Processing ......................................................... 249
    Object-Oriented Concepts .................................................... 250

PART 4  ODS Styles Reference 257

Chapter 13 • Overview ............................................................. 259
    Understanding Styles, Style Elements, and Style Attributes ........ 259
    Using Styles with Base SAS Procedures .................................. 265

Chapter 14 • Style Templates ....................................................... 269
    Viewing ODS Styles Supplied by SAS .................................... 269
    Table of Suggested ODS Styles ........................................... 271
    Program for Viewing Multiple Styles .................................... 272
    ODS Styles Gallery ............................................................... 274

Chapter 15 • Style Elements ....................................................... 311
    General ODS Style Elements ............................................... 311
    Style Elements Affecting Template-Based Graphics .................. 321
    Style Elements Affecting Device-Based Graphics ..................... 329

Chapter 16 • Style Attributes ...................................................... 339
    Overview ............................................................................. 339
    Style Attributes Tables ......................................................... 339
    Style Attributes Detailed Information .................................... 350
Contents
About This Book

Lua License


Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the “Software”), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions: The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Syntax Conventions for the SAS Language

Overview of Syntax Conventions for the SAS Language

SAS uses standard conventions in the documentation of syntax for SAS language elements. These conventions enable you to easily identify the components of SAS syntax. The conventions can be divided into these parts:

- syntax components
- style conventions
- special characters
- references to SAS libraries and external files
**Syntax Components**

The components of the syntax for most language elements include a keyword and arguments. For some language elements, only a keyword is necessary. For other language elements, the keyword is followed by an equal sign (=). The syntax for arguments has multiple forms in order to demonstrate the syntax of multiple arguments, with and without punctuation.

**Keyword**

specifies the name of the SAS language element that you use when you write your program. Keyword is a literal that is usually the first word in the syntax. In a CALL routine, the first two words are keywords.

In these examples of SAS syntax, the keywords are bold:

- CHAR *(string, position)*
- CALL RANBIN *(seed, n, p, x)*;
- ALTER *(alter-password)*
- BEST w.
- REMOVE <data-set-name>*

In this example, the first two words of the CALL routine are the keywords:

CALL RANBIN*(seed, n, p, x)*

The syntax of some SAS statements consists of a single keyword without arguments:

- DO;

  ... SAS code ...

- END;

Some system options require that one of two keyword values be specified:

- DUPLEX | NODUPLEX

Some procedure statements have multiple keywords throughout the statement syntax:

- CREATE <UNIQUE> INDEX index-name ON table-name *(column-1 <, column-2, ...>)*

**Argument**

specifies a numeric or character constant, variable, or expression. Arguments follow the keyword or an equal sign after the keyword. The arguments are used by SAS to process the language element. Arguments can be required or optional. In the syntax, optional arguments are enclosed in angle brackets (*<*>*).

In this example, *string* and *position* follow the keyword CHAR. These arguments are required arguments for the CHAR function:

- CHAR *(string, position)*

Each argument has a value. In this example of SAS code, the argument *string* has a value of ‘summer’, and the argument *position* has a value of 4:

x=char(‘summer’, 4);

In this example, *string* and *substring* are required arguments, whereas *modifiers* and *startpos* are optional.

- FIND*(string, substring <, modifiers> <, startpos>)*
argument(s)
specifies that one argument is required and that multiple arguments are allowed. Separate arguments with a space. Punctuation, such as a comma (, ) is not required between arguments.

The MISSING statement is an example of this form of multiple arguments:

MISSING character(s);

<LITERAL_ARGUMENT> argument-1 <<LITERAL_ARGUMENT> argument-2 ...>
specifies that one argument is required and that a literal argument can be associated with the argument. You can specify multiple literals and argument pairs. No punctuation is required between the literal and argument pairs. The ellipsis (...) indicates that additional literals and arguments are allowed.

The BY statement is an example of this argument:

BY <DESCENDING> variable-1 <<DESCENDING> variable-2 ...>;

argument-1 <option(s)> <argument-2 <option(s)> ...

specifies that one argument is required and that one or more options can be associated with the argument. You can specify multiple arguments and associated options. No punctuation is required between the argument and the option. The ellipsis (...) indicates that additional arguments with an associated option are allowed.

The FORMAT procedure PICTURE statement is an example of this form of multiple arguments:

PICTURE name <(format-option(s))>
<value-range-set-1 <(picture-1-option(s))>
<value-range-set-2 <(picture-2-option(s))> ...>>;

argument-1=value-1 <argument-2=value-2 ...

specifies that the argument must be assigned a value and that you can specify multiple arguments. The ellipsis (...) indicates that additional arguments are allowed. No punctuation is required between arguments.

The LABEL statement is an example of this form of multiple arguments:

LABEL variable-1=label-1 <variable-2=label-2 ...>;

argument-1 <, argument-2, ...

specifies that one argument is required and that you can specify multiple arguments that are separated by a comma or other punctuation. The ellipsis (...) indicates a continuation of the arguments, separated by a comma. Both forms are used in the SAS documentation.

Here are examples of this form of multiple arguments:

AUTHPROVIDERDOMAIN (provider-1:domain-1 <, provider-2:domain-2, ...>
INTO :macro-variable-specification-1 <, :macro-variable-specification-2, ...>

Note: In most cases, example code in SAS documentation is written in lowercase with a monospace font. You can use uppercase, lowercase, or mixed case in the code that you write.

**Style Conventions**

The style conventions that are used in documenting SAS syntax include uppercase bold, uppercase, and italic: 
UPPERCASE BOLD
identifies SAS keywords such as the names of functions or statements. In this example, the keyword ERROR is written in uppercase bold:

```
ERROR <message>;
```

UPPERCASE
identifies arguments that are literals.

In this example of the CMPMODEL= system option, the literals include BOTH, CATALOG, and XML:

```
CMPMODEL= BOTH | CATALOG | XML |
```

italic
identifies arguments or values that you supply. Items in italic represent user-supplied values that are either one of the following:

- nonliteral arguments. In this example of the LINK statement, the argument label is a user-supplied value and therefore appears in italic:

```
LINK label;
```

- nonliteral values that are assigned to an argument.

In this example of the FORMAT statement, the argument DEFAULT is assigned the variable default-format:

```
FORMAT variable(s) <format> <DEFAULT = default-format>;
```

Special Characters

The syntax of SAS language elements can contain the following special characters:

- an equal sign identifies a value for a literal in some language elements such as system options.

In this example of the MAPS system option, the equal sign sets the value of MAPS:

```
MAPS= location-of-maps
```

- angle brackets identify optional arguments. A required argument is not enclosed in angle brackets.

In this example of the CAT function, at least one item is required:

```
CAT (item-1 <, item-2, …>)
```

- a vertical bar indicates that you can choose one value from a group of values. Values that are separated by the vertical bar are mutually exclusive.

In this example of the CMPMODEL= system option, you can choose only one of the arguments:

```
CMPMODEL= BOTH | CATALOG | XML
```

- an ellipsis indicates that the argument can be repeated. If an argument and the ellipsis are enclosed in angle brackets, then the argument is optional. The repeated argument must contain punctuation if it appears before or after the argument.
In this example of the CAT function, multiple *item* arguments are allowed, and they must be separated by a comma:

```
CAT (item-1 <, item-2, ...>)
```

'value' or "value"

indicates that an argument that is enclosed in single or double quotation marks must have a value that is also enclosed in single or double quotation marks.

In this example of the FOOTNOTE statement, the argument *text* is enclosed in quotation marks:

```
FOOTNOTE<n> <ods-format-options 'text' | "text">;
```

a semicolon indicates the end of a statement or CALL routine.

In this example, each statement ends with a semicolon:

```
data namegame;
  length color name $8;
  color = 'black';
  name = 'jack';
  game = trim(color) || name;
run;
```

**References to SAS Libraries and External Files**

Many SAS statements and other language elements refer to SAS libraries and external files. You can choose whether to make the reference through a logical name (a libref or fileref) or use the physical filename enclosed in quotation marks. If you use a logical name, you typically have a choice of using a SAS statement (LIBNAME or FILENAME) or the operating environment's control language to make the reference. Several methods of referring to SAS libraries and external files are available, and some of these methods depend on your operating environment.

In the examples that use external files, SAS documentation uses the italicized phrase *file-specification*. In the examples that use SAS libraries, SAS documentation uses the italicized phrase *SAS-library* enclosed in quotation marks:

```
infile file-specification obs = 100;
libname libref 'SAS-library';
```
Part 1

Advanced PROC DOCUMENT Topics

Chapter 1
Introduction ......................................................... 3

Chapter 2
Restructuring Output with PROC DOCUMENT .................. 5

Chapter 3
Using WHERE Expressions in PROC DOCUMENT .............. 21

Chapter 4
Working with Output Objects and Table Templates .......... 29
Chapter 1
Introduction

Advanced PROC DOCUMENT Features

The ODS DOCUMENT destination enables you to store the output objects from your procedure or process in a special type of item store called a document store. With the DOCUMENT procedure, you are not limited to simply regenerating the same report. You can change the order in which objects are rendered, the table of contents, the templates that are used, macro variables, and ODS and system options. The following sections show you some of the more advanced features of PROC DOCUMENT.

- For information about rearranging your output with PROC DOCUMENT, see Chapter 2, “Restructuring Output with PROC DOCUMENT,” on page 5.
- For information about using WHERE expressions with PROC DOCUMENT, see Chapter 3, “Using WHERE Expressions in PROC DOCUMENT,” on page 21.
- For information about using PROC DOCUMENT to modify output objects, see Chapter 4, “Working with Output Objects and Table Templates,” on page 29.
Chapter 2
Restructuring Output with PROC DOCUMENT

Overview

The DOCUMENT procedure enables you to restructure your report in unique ways. You can rename or create folders, include or exclude output objects, and rename output objects. Restructuring your report is one way to create a customized table of contents for destinations such as HTML and PDF. You can use the Documents window or PROC DOCUMENT syntax to modify your output.

Viewing an ODS Document in the Documents Window

To view the Documents window, submit this command in the command bar:

```plaintext
odsdocuments
```

The following display shows the Documents window that contains the ODS document named `Sasuser.Univ`. In the display, notice that `Sasuser.Univ` contains several directory levels. The `Exponential_x` directory contains the `Exp` output object. When you double-click an output object, such as `Exp`, that output object is replayed in the Results window to all open destinations.
A Documents window contains these items:

- **entry**
  - is an output object, link, or directory.
  
  *Note:* Only output objects of the type Document are displayed in the Documents window.

- **directory**
  - is a grouping of ODS document entries.

- **link**
  - is a symbolic link from one specified output object to another output object.
  
  *Note:* Within the Documents window, a link is called a shortcut.

- **ODS document**
  - is the name of an ODS document.

---

**Viewing the Contents of an ODS Document**

After you have created a document with the ODS DOCUMENT statement, you can use PROC DOCUMENT to view the contents of your document. You can rearrange, duplicate, or remove output from the results of a procedure or a database query without invoking the procedures from the original report. The first step is to view your document’s contents by using the LIST statement. The LIST statement enables you to look at the object list and folder structure within the ODS document. The following code creates a list of all levels of the document Work.Prddoc:

```sas
proc document name=work.prddoc;
  list / levels=all;
run;
quit;
```

The LIST statement can be used to list what is in an entire document or just one of the entries. For more information about the LIST statement, see “LIST Statement” in *SAS Output Delivery System: Procedures Guide*. 
In the following display, every folder icon in the Results window corresponds to an item with a type of "Dir" in the LIST statement output. Every table created by a procedure corresponds to an item with a type of "Table" in the LIST statement output.

**Figure 2.2  PROC DOCUMENT List Output Compared to Results Window**

Using the Documents Window Pop-up Menu

The Documents window has a pop-up menu with features that are also available through batch processing. You can create a new, modified ODS document by using the Documents window. This enables you to change the table of contents in the PDF and HTML destination.

To create a new document, follow these steps:

1. Type `odsdocuments` in the command bar. The Documents window appears.

2. Create a new ODS document by right-clicking the **Documents** folder at the top of the Documents window and selecting **New Document** from the menu.
3. When the New Document window appears, select a library to store the new document and provide a name for the document (NEWPRD, for example). Then click **OK**.

   *Figure 2.3 Naming a New Document*

4. Create a folder in the new document by right-clicking the folder that represents the new document and selecting **New Folder**.

5. Enter the country and the description in the New Folder window, and then select **OK**. If a **Description** value is provided, that value is used for the folder name. Note that in the following figure, United States is used as the folder name, not USA.

   *Figure 2.4 Name the New Folder*
The folder with all three countries is shown here. Even though the folders are expanded, they have no objects, so there is nothing to show in the expanded view.

**Figure 2.5  New Document with New Folders**

6. You can either copy and paste or drag and drop from the original document store (Work.Prddoc) to the new document, Work.Newprd.

**Figure 2.6  Moving Output Objects**

7. You can also rename the output objects. Right-click the output object that you want to rename, and select **Rename** from the menu.
In the following figure, each “Table 1” has been renamed “Predicted Sales”, which is the value entered in the Description field.

**Figure 2.7 Renaming Output Objects**

The following figure shows the final structure of the Work.Newprd document.

**Figure 2.8 The Work.Newprd Document**

8. To replay the entire document, right-click the document and select **Open As** from the drop-down menu.

   When the Open As window appears, select a destination, such as PDF (highlighted), and then click **OK**. You can select multiple destinations by pressing the Ctrl key and selecting the destinations that you want.
Much of what can be done in this interactive mode to rearrange and rename output objects can be done with PROC DOCUMENT syntax. The PROC DOCUMENT syntax actually parallels the process that you go through in the interactive window. This means that study of the interactive Document window enhances your understanding of the PROC DOCUMENT syntax for accomplishing the same end results.

Although this is an easy way to create a document, if you replay your original procedure steps, you have to repeat the restructuring process. You can use the corresponding PROC DOCUMENT code to re-create your document, or you can use links to output objects to create your document. For an example of a method that uses links to output objects to restructure your output, see “Example 2: Creating Custom Output with Links” on page 15.

Examples

Example 1: Restructuring Output

<table>
<thead>
<tr>
<th>Features:</th>
<th>PROC DOCUMENT statements</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>DOC</td>
</tr>
<tr>
<td></td>
<td>DIR</td>
</tr>
<tr>
<td></td>
<td>COPY TO</td>
</tr>
<tr>
<td></td>
<td>MAKE</td>
</tr>
<tr>
<td></td>
<td>PROC DOCUMENT</td>
</tr>
<tr>
<td></td>
<td>REPLAY</td>
</tr>
<tr>
<td></td>
<td>SETLABEL</td>
</tr>
</tbody>
</table>

| Other features: | ODS PDF statement |
ODS DOCUMENT statement

ODS SELECT statement

Details

PROC DOCUMENT allows you to create a new, modified ODS document in the Documents window or in a SAS program. You can use PROC DOCUMENT syntax to accomplish the same tasks shown in the section “Using the Documents Window Pop-up Menu” on page 7. Using the Documents window is useful when you have a few documents that you need to create and replay. However, if you use SAS Enterprise Guide or SAS Studio (which do not have the Documents window) or you want to create an ODS document store as part of nightly production processing, then the PROC DOCUMENT syntax enables you to accomplish these creation, management, and replay tasks. The following SAS program corresponds to the actions that were done in the Documents window.

The following code creates the ODS document Work.Prddoc, which we are working with. The ODS SELECT statement specifies that only the output ExtremeObs (which is one of the output objects created by PROC UNIVARIATE) is added to the document. All of the output objects created by PROC TABULATE are added to the document.

```
ods html close;
proc sort data=sashelp.prdsale out=prdsale;
   by Country;
run;
ods document name=work.prddoc(write);
proc tabulate data=prdsale;
   by Country;
   var predict;
   class prodttype;
   table prodttype all,
        predict*(min mean max);
run;
ods select ExtremeObs;
proc univariate data=prdsale;
   by Country;
   var actual;
run;
ods document close;
```

Program

```
ods html close;
proc document;
   doc name=Work.Newprd(WRITE);
   dir \\
   make USA;
   setlabel USA 'United States';
   copy \Work.Prddoc\Univariate#1\ByGroup3#1\ACTUAL#1\ExtremeObs#1 TO USA#1;
   copy \Work.Prddoc\Tabulate#1\ByGroup3#1\Report#1\Table#1 TO USA#1;
   setlabel USA#1\Table#1 'Predicted Sales';

   make Germany;
   setlabel Germany 'Germany';
```
Example 1: Restructuring Output

```
copy \Work.Prddoc\Univariate#1\ByGroup2#1\ACTUAL#1\ExtremeObs#1 TO Germany#1;
copy \Work.Prddoc\Tabulate#1\ByGroup2#1\Report#1\Table#1 TO Germany#1;
setlabel Germany#1\Table#1 'Predicted Sales';

make Canada;
setlabel Canada 'Canada';
copy \Work.Prddoc\Univariate#1\ByGroup1#1\ACTUAL#1\ExtremeObs#1 TO Canada#1;
copy \Work.Prddoc\Tabulate#1\ByGroup1#1\Report#1\Table#1 TO Canada#1;
setlabel Canada#1\Table#1 'Predicted Sales';

quit;
ods pdf file='CustomPrd.pdf' contents=yes;
proc document name=work.Newprd;
   replay;
run;
quit;
ods _all_ close;
```

Program Description

**Close the HTML destination.** The HTML destination is open by default. Closing the HTML destination saves system resources.
```
ods html close;
```

**Create the ODS document Newprd.** The DOC statement with the WRITE option specified creates a new document in the Work directory.
```
proc document;
   doc name=Work.Newprd(WRITE);
dir \;
```

**Create a new folder and copy output objects into it.** The MAKE statement creates the folder "USA". The SETLABEL statement specifies the text of the label. The COPY TO statements copy the output objects ExtremeObs#1 and Table#1 to the new folder.
```
make USA;
setlabel USA 'United States';
copy \Work.Prddoc\Univariate#1\ByGroup3#1\ACTUAL#1\ExtremeObs#1 TO USA#1;
copy \Work.Prddoc\Tabulate#1\ByGroup3#1\Report#1\Table#1 TO USA#1;
setlabel USA#1\Table#1 'Predicted Sales';
```

**Create a new folder and copy output objects into it.** The MAKE statement creates the folder "Germany". The SETLABEL statement specifies the text of the label. The COPY TO statements copy the output objects ExtremeObs#1 and Table#1 to the new folder.
```
make Germany;
setlabel Germany 'Germany';
copy \Work.Prddoc\Univariate#1\ByGroup2#1\ACTUAL#1\ExtremeObs#1 TO Germany#1;
copy \Work.Prddoc\Tabulate#1\ByGroup2#1\Report#1\Table#1 TO Germany#1;
setlabel Germany#1\Table#1 'Predicted Sales';
```
Create a new folder and copy output objects into it. The MAKE statement creates the folder "Canada". The SETLABEL statement specifies the text of the label. The COPY TO statements copy the output objects ExtremeObs#1 and Table#1 to the new folder.

```plaintext
make Canada;
setlabel Canada 'Canada';
copy \\Work.Prddoc\Univariate#1\ByGroup1#1\ACTUAL#1\ExtremeObs#1 TO Canada#1;
copy \\Work.Prddoc\Tabulate#1\ByGroup1#1\Report#1\Table#1 TO Canada#1;
setlabel Canada#1\Table#1 'Predicted Sales';
```

End the DOCUMENT procedure.

```plaintext
quit;
```

Specify the PDF destination and replay the document. The ODS PDF statement opens the PDF destination and creates the output file. The REPLAY statement in the PROC DOCUMENT step replays the document Newprd to the PDF destination.

```plaintext
ods pdf file='CustomPrd.pdf' contents=yes;
proc document name=work.Newprd;
  replay;
run;
quit;
ods _all_ close;
```

PDF Output

Output 2.1 Original PDF Table of Contents
Example 2: Creating Custom Output with Links

Features: PROC DOCUMENT statements
- DOC
- DIR
- LINK TO
- MAKE
- PROC DOCUMENT
- REPLAY
- SETLABEL

Other features: ODS PDF statement
- ODS DOCUMENT statement
- ODS SELECT statement

Details

The following example creates a new a document with links to the output objects in the original document. You have to create the custom structure with links only once, and then you can replay the document any number of times. This prevents you from having to make the custom structure every time you run SAS.

In the example “Example 1: Restructuring Output” on page 11, you learned how to rearrange output with the Documents window. However, you can accomplish the same modifications by using links in a new document.

The following code creates the ODS document Work.Prddoc, which we are working with.

```sas
ods listing close;
proc sort data=sashelp.prdsale out=prdsale;
   by Country;
run;
ods document name=work.prddoc(write);
proc tabulate data=prdsale;
   by Country;
   var predict;
   class prodtype;
   table prodtype all,
       predict*(min mean max);
run;
```
ods select ExtremeObs;
proc univariate data=prdsale;
   by Country;
   var actual;
run;
ods document close;

Program
ods html cose;
proc document;
   doc name=Work.NewprdLinks(WRITE);
   dir \\
   make USA;
   setlabel USA 'United States';
   link \Work.Prddoc\Univariate#1\ByGroup3#1\ACTUAL#1\ExtremeObs#1 TO USA#1;
   link \Work.Prddoc\Tabulate#1\ByGroup3#1\Report#1\Table#1 TO USA#1;
   setlabel USA#1\Table#1 'Predicted Sales';
make Germany;
   setlabel Germany 'Germany';
   link \Work.Prddoc\Univariate#1\ByGroup2#1\ACTUAL#1\ExtremeObs#1 TO Germany#1;
   link \Work.Prddoc\Tabulate#1\ByGroup2#1\Report#1\Table#1 TO Germany#1;
   setlabel Germany#1\Table#1 'Predicted Sales';
make Canada;
   setlabel Canada 'Canada';
   link \Work.Prddoc\Univariate#1\ByGroup1#1\ACTUAL#1\ExtremeObs#1 TO Canada#1;
   link \Work.Prddoc\Tabulate#1\ByGroup1#1\Report#1\Table#1 TO Canada#1;
   setlabel Canada#1\Table#1 'Predicted Sales';
quit;
ods pdf file='CustomPrdLinks.pdf' contents=yes;
proc document name=work.NewprdLinks;
   replay;
run;
quit;

Program Description

Close the HTML destination. The HTML destination is open by default. Closing the HTML destination saves system resources.
ods html cose;

Create the ODS document NewprdLinks. The DOC statement with the WRITE option specified creates a new document in the Work directory.
proc document;
   doc name=Work.NewprdLinks(WRITE);
   dir \\

Create a new folder and create links to the original output objects. The MAKE statement creates the folder "USA". The SETLABEL statement specifies the text of the
label. The LINK TO statements copy the output objects ExtremeObs#1 and Table#1 to
the new folder.

make USA;
setlabel USA 'United States';
link \Work.Prddoc\Univariate#1\ByGroup3#1\ACTUAL#1\ExtremeObs#1 TO USA#1;
link \Work.Prddoc\Tabulate#1\ByGroup3#1\Report#1\Table#1 TO USA#1;
setlabel USA#1\Table#1 'Predicted Sales';

Create a new folder and create links to the original output objects. The MAKE
statement creates the folder "Germany". The SETLABEL statement specifies the text of
the label. The LINK TO statements copy the output objects ExtremeObs#1 and Table#1
to the new folder.

make Germany;
setlabel Germany 'Germany';
link \Work.Prddoc\Univariate#1\ByGroup2#1\ACTUAL#1\ExtremeObs#1 TO Germany#1;
link \Work.Prddoc\Tabulate#1\ByGroup2#1\Report#1\Table#1 TO Germany#1;
setlabel Germany#1\Table#1 'Predicted Sales';

Create a new folder and create links to the original output objects. The MAKE
statement creates the folder "Canada". The SETLABEL statement specifies the text of
the label. The LINK TO statements copy the output objects ExtremeObs#1 and Table#1
to the new folder.

make Canada;
setlabel Canada 'Canada';
link \Work.Prddoc\Univariate#1\ByGroup1#1\ACTUAL#1\ExtremeObs#1 TO Canada#1;
link \Work.Prddoc\Tabulate#1\ByGroup1#1\Report#1\Table#1 TO Canada#1;
setlabel Canada#1\Table#1 'Predicted Sales';

End the DOCUMENT procedure.

quit;

Specify the PDF destination and replay the document. The ODS PDF statement
opens the PDF destination and creates the output file. The REPLAY statement in the
PROC DOCUMENT step replays the document NewprdLinks to the PDF destination.

ods pdf file='CustomPrdLinks.pdf' contents=yes;
proc document name=work.NewprdLinks;
  replay;
run;
quit;

Document Window Structure
In the Documents window, the following icon indicates that the output object is a link to
an output object:
Output 2.3  ODS Document Newpdrlinks Shown in the Documents Window

PDF Output
The PDF table of contents and output is exactly the same as in “Example 1: Restructuring Output” on page 11.

Output 2.4  Original PDF Table of Contents

Table of Contents

The Tabulate Procedure
Country=CANADA ........................................... 1
Cross-tabular summary report ................................ 1
Table 1 .......................................................... 1

Country=GERMANY .......................................... 2
Cross-tabular summary report ................................ 2
Table 1 .......................................................... 2

Country=U.S.A ............................................... 3
Cross-tabular summary report ................................ 3
Table 1 .......................................................... 3

The Univariate Procedure
Country=CANADA ........................................... 4

ACTUAL ...................................................... 4

Extreme Observations
Country=GERMANY .......................................... 5

ACTUAL ...................................................... 5

Extreme Observations
Country=U.S.A ............................................... 6

ACTUAL ...................................................... 6

Extreme Observations
Output 2.5  PDF Table of Contents After the Output Has Been Modified

<table>
<thead>
<tr>
<th>Table of Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>United States</td>
</tr>
<tr>
<td>Extreme Observations</td>
</tr>
<tr>
<td>Predicted Sales</td>
</tr>
<tr>
<td>Germany</td>
</tr>
<tr>
<td>Extreme Observations</td>
</tr>
<tr>
<td>Predicted Sales</td>
</tr>
<tr>
<td>Canada</td>
</tr>
<tr>
<td>Extreme Observations</td>
</tr>
<tr>
<td>Predicted Sales</td>
</tr>
</tbody>
</table>
Chapter 3
Using WHERE Expressions in PROC DOCUMENT

Overview
WHERE expressions in PROC DOCUMENT statements enable you to conditionally perform actions on objects within the document. For example, you can use a WHERE expression with a subsetting variable to help you find common metadata values in ODS documents. You can also use WHERE expressions to only print output objects with specific BY variable values.

WHERE Expression Syntax
WHERE expressions can be used in the following PROC DOCUMENT statements:
- COPY TO
- DELETE
- LINK
- LIST
- MOVE TO
- Replay

For complete documentation about WHERE expressions with any of the preceding statements, see the statement documentation in “The DOCUMENT Procedure” in SAS Output Delivery System: Procedures Guide.

Document paths can be followed by a WHERE clause to further subset the objects matched by a path. The general form of WHERE clause usage is shown below:

document-path(WHERE=(where-expression))

(WHERE=(where-expression-1<operator >))

conditionally selects a subset of entries in an ODS document.
where-expression
is an arithmetic or logical expression that consists of a sequence of operators and
operands.

operand
is one of the following:

constant
is a fixed value such as a date literal, a value, or a BY variable value.

SAS function
For information about SAS functions, see SAS Functions and CALL
Routines: Reference.

subsetting variable
is a special type of WHERE expression operand used by the
DOCUMENT procedure to help you find common metadata values in
ODS documents. Variables available in a WHERE clause are listed in the
following table.

Table 3.1 Table of WHERE Clause Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>CDATE</em></td>
<td>Creation date of the item</td>
<td>The following MOVE TO statement moves all entries of the type 'Graph' with a creation date of 16JUL2004 to the Monthly directory in Work.MyDoc: move ^(where=(<em>type</em> = 'Graph' and <em>cdate</em> = '16JUL2004'd)) to \ work.mydoc\monthly; run;</td>
</tr>
<tr>
<td><em>CDATETIME</em></td>
<td>Creation datetime of the item</td>
<td>The following COPY TO statement copies all entries with a creation datetime of May 1, 2003, at 9:30 to the Monthly directory of Work.MyDoc: copy ^(where=(<em>cdatetime</em> = '01may04:9:30:00'dt)) to \ work.mydoc\monthly; run;</td>
</tr>
<tr>
<td><em>CTIME</em></td>
<td>Creation time of the item</td>
<td>The following DELETE statement deletes all entries with a creation time of 9:25:19 PM: delete ^(where=(<em>ctime</em> = '9:25:19pm't)); run;</td>
</tr>
<tr>
<td><em>LABEL</em></td>
<td>Item label</td>
<td>The following LIST statement lists all tables containing the label 'Type III Model' within the GLM procedure: list glm(where=(<em>type</em> = 'table' and <em>label</em> ? 'Type III Model')); run;</td>
</tr>
<tr>
<td><em>LABELPATH</em></td>
<td>Labels of all entries in the path</td>
<td>The following LIST statement lists all items containing “Fit Statistics” in the label path: list gml(where=(<em>labelpath</em> ? &quot;Fit Statistics&quot;) / levels=all; run;</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
<td>Example</td>
</tr>
<tr>
<td>----------</td>
<td>------------------------------------</td>
<td>-------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| _MAX_    | The last observation               | The following REPLAY statement replays all observations except the last observation:  
|          |                                    | ```replay class(where=(_obs_ < _max_));```                             |
| _MDATE_  | Modification date of the item      | The following MOVE TO statement moves all entries of the type 'Graph' with a modification date of 16JUL2004 to the Monthly directory of Work.MyDoc:  
|          |                                    | ```move *(where=(_type_ = 'Graph' and _mdate_ = '16JUL2004'd)) to \work.mydoc\monthly; run;``` |
| _MDATETIME_ | Modification datetime of the item | The following REPLAY statement replays all entries with a modification datetime of May 1, 2003, at 9:30:  
|          |                                    | ```replay *(where=(_mdatetime_ = '01may04:9:30:00'dt)); run;```          |
| _MIN_    | The first observation              | The following REPLAY statement replays all observations except the first observation:  
|          |                                    | ```replay class(where=(_obs_ > _min_));```                            |
| _MTIME_  | Modification time of the item      | The following COPY TO statement copies all entries with a modification time of 9:25:19 PM to the Monthly directory of Work.MyDoc:  
|          |                                    | ```copy *(where=(_mtime_ = '9:25:19pm't)) to \work.mydoc\monthly; run;``` |
| _NAME_   | Item name                          | The following DELETE statement deletes all entries that contain the name “stemleng” within the GLM procedure:  
|          |                                    | ```delete glm(where=|_name_ ? 'stemleng' |);```                        |
| _OBS_    | The current observation number in an output object | The following REPLAY statement replays all but the first ten observations:  
|          |                                    | ```replay class(where=(_obs_ > 10))```                                 |
| observation-number | The observation number to be replayed | The following REPLAY statement replays the first, third, fifth, seventh, and ninth observations:  
|          |                                    | ```replay class(where=(_obs_ in {1,3,5,7,9}))```                        |
| observation-variable   | The name of an observation          | The following REPLAY statement replays all observations where the variable Weight is greater than 100:  
|          |                                    | ```replay class(where=(weight>100));```                              |
### Example

**_PATH_**

Item path

The following LIST statement lists all entries with a path containing the substring 'Anova' at all levels of the current directory:

```plaintext
list ^(where=(_path_ ? 'Anova'));
run;
```

**_SEQNO_**

Item sequence number

The following REPLAY statement replays all entries that have a sequence number of 2 in the GLM procedure:

```plaintext
replay glm(where=(_seqno_ = 2));
```

**_TYPE_**

Item type

The following MOVE TO statement moves all entries of the type 'Graph' with a creation date of July 16, 2004, to the Monthly directory of Work.MyDoc:

```plaintext
move ^(where=(_type_ = 'Graph' and _cdate_ = '16JUL2004'd)) to \work.mydoc\monthly;
run;
```

**variable**

BY variable

The following MOVE TO statement moves all entries where the value of the variable Gender is 'F' to the Monthly directory of Work.MyDoc:

```plaintext
move ^(where=(gender='F')) to \work.mydoc\monthly;
run;
```

### Comparison Operators

*operator* compares one variable with a value or another variable. *operator* can be AND, OR NOT, OR, AND NOT, or a comparison operator.

#### Table 3.2 Comparison Operators

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Mnemonic Equivalent</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>=</td>
<td>EQ</td>
<td>Equal to</td>
</tr>
<tr>
<td>^= or -= or == or &lt;&gt;</td>
<td>NE</td>
<td>Not equal to</td>
</tr>
<tr>
<td>&gt;</td>
<td>GT</td>
<td>Greater than</td>
</tr>
<tr>
<td>&lt;</td>
<td>LT</td>
<td>Less than</td>
</tr>
<tr>
<td>&gt;=</td>
<td>GE</td>
<td>Greater than or equal to</td>
</tr>
<tr>
<td>&lt;=</td>
<td>LE</td>
<td>Less than or equal to</td>
</tr>
<tr>
<td>IN</td>
<td></td>
<td>Equal to one from a list of values</td>
</tr>
</tbody>
</table>
Example: Opening and Listing ODS Documents Using WHERE Expressions

**Features:**
- PROC DOCUMENT statements
  - PROC DOCUMENT
  - LIST statement
  - REPLAY statement

**Other features:**
- ODS PDF statement
- ODS DOCUMENT statement
- PROC UNIVARIATE

**Data set:**
- DistrData

**Note:** See “Creating the Univ ODS Document” on page 391 for the SAS code that creates the Univ ODS document.

**Details**
This example shows you how to do these tasks:
- open an ODS document
- replay a table and send the output to the LISTING and PDF destinations
- list specific entries in an ODS document by using WHERE expressions
- list the details of a specified entry
- replay an ODS document to a PDF file

**Program**
```sas
options nodate nonumber;
odspath=file='your_file.pdf';
proc document name=univ;
list ^ (where=( _type_ = 'Graph' or _type_ = 'Table' ) ) / levels=all;
replay univariate#1\Normal_x#1\Histogram#1\Histogram#1;
list \Work.Univ\Univariate#2\Exponential_x#1\Histogram#1\Exponential#1\FitQuantiles#1 / details;
replay \Work.Univ\Univariate#2\Exponential_x#1\Histogram#1\Exponential#1\FitQuantiles#1;
run;
quit;
odspath close;
```

**Program Description**

**Set the SAS system options.** The NODATE option suppresses the display of the date and time in the output. The NONUMBER option suppresses the printing of page numbers.
options nodate nonumber;

Specify that you want to replay the output to a PDF file. The ODS PDF statement opens the PDF file that contains the replayed histogram. The FILE= statement sends all output objects to the external file that you specify. HTML output is created by default.

    ods pdf file='your_file.pdf';

Open the ODS document Work.Univ. The PROC DOCUMENT statement with the NAME= option specified opens the ODS document Work.Univ for updates. WORK.Univ was created in the example “Navigating the directory and Listing the Entries”.

    proc document name=univ;

List the entries that are associated with the current document and replay a histogram. By using a WHERE expression, the LIST statement lists only entries that are graphs or tables. The LEVELS=ALL option specifies that detailed information about all levels be shown. The ^ symbol represents the current directory. The REPLAY statement replays the Histogram#1 entry to all open ODS destinations.

    list ^ (where=(type_ = 'Graph' or _type_ = 'Table')) / levels=all;
    replay univariate#1\Normal_x#1\Histogram#1\Histogram#1;

List the details of the FitQuantiles table, and replay the FitQuantiles table. The LIST statement with the DETAILS option specifies the listing of the properties of the entry FitQuantiles table. The REPLAY statement replays FITQUANTILES to open destinations.

    list \\Work.Univ\Univariate#2\Exponential_x#1\Histogram#1\Exponential#1\FitQuantiles#1
       / details;
    replay \\Work.Univ\Univariate#2\Exponential_x#1\Histogram#1\Exponential#1\FitQuantiles#1;
run;

Terminate the DOCUMENT procedure and close the PDF destination. Specify the QUIT statement to terminate the DOCUMENT procedure. If you omit QUIT, then you cannot view DOCUMENT procedure output. The ODS PDF CLOSE statement closes the PDF destination and all the files that are associated with it. If you do not close the destination, then you cannot view the files.

    quit;
    ods pdf close;
Output

This display is page 1 of the ODS document Work.Univ that was sent to the PDF destination. You can browse the output by clicking the bookmarks.

Output 3.1  List of the Graphs and Tables Found in Work.Univ, Viewed in Acrobat Reader

Output 3.2  Replayed Normal Distribution Histogram
Output 3.3  Details of the FitQuantiles#1 Table

Output 3.4  Replayed FitQuantiles#1 Table
Chapter 4
Working with Output Objects and Table Templates

Overview
PROC DOCUMENT enables you to view information about the output objects that are stored in a document. When a document is created, a reference to the template that output objects use is created. Templates are stored in an ODS document only when you use PROC PRINT, PROC REPORT, PROC TABULATE, or the ODS Graphics procedures. PROC DOCUMENT enables you to view specific information about the output objects' template in your document. You can then use this information to further customize your output by specifying the template that you want to use.

Displaying the Template Used by Output Objects
You can use the OBTEMPL statement to display the template of an output object. The OBTEMPL statement has the following form:

```
OBTEMPL output-object;
```

`output-object` specifies the pathname of the output object.

By viewing the output object's table template, you can view the name of the table template that is used for your output and the table attributes that are used to customize the output. For graphics, you can view the name of the graphic template that is used for a graphic. It is important to know the name of the default table template so that you can create a customized version. You can use either the TEMPLATE procedure or the ODSTABLE procedure to customize table templates. For more information about the TEMPLATE procedure and the ODSTABLE procedure, see *SAS Output Delivery System: Procedures Guide.*
Example: Using PROC DOCUMENT to Work with Table Templates

**Features:**
- PROC DOCUMENT statements
  - DIR
  - LIST statement
  - MAKE
  - OBTEMPL
  - PROC DOCUMENT
  - REPLAY
  - SETLABEL

**Other features:**
- ODS DOCUMENT statement
- PROC ODSTABLE
- PROC SQL
- PROC TEMPLATE statements
  - DELETE
  - CELLSYLETE AS

**Details**

In this example, the PROC SQL steps create two tables. The following output shows the default output for the PROC SQL steps:

**Output 4.1  Default PROC SQL Output**

<table>
<thead>
<tr>
<th>Make</th>
<th>Model</th>
<th>MPG (City)</th>
<th>MPG (Highway)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Honda</td>
<td>Civic Hybrid 4dr manual (gas/electric)</td>
<td>46</td>
<td>51</td>
</tr>
<tr>
<td>Honda</td>
<td>Insight 2dr (gas/electric)</td>
<td>60</td>
<td>66</td>
</tr>
<tr>
<td>Honda</td>
<td>Civic HX 2dr</td>
<td>36</td>
<td>44</td>
</tr>
<tr>
<td>Toyota</td>
<td>Prius 4dr (gas/electric)</td>
<td>59</td>
<td>51</td>
</tr>
<tr>
<td>Toyota</td>
<td>Echo 2dr manual</td>
<td>35</td>
<td>43</td>
</tr>
<tr>
<td>Toyota</td>
<td>Echo 2dr manual</td>
<td>35</td>
<td>43</td>
</tr>
</tbody>
</table>
Suppose that you want to modify the look of these tables. You can use PROC ODSTABLE to create customized table templates. You can then use PROC DOCUMENT to apply the customized table templates to your output objects. You can even create two different table templates with PROC ODSTABLE and apply each one to a different output object with PROC DOCUMENT.

**Create the ODS Document and List the Output Objects**

```
ods document name=SqlTempl(write);
proc sql;
title "Cars with City MPG Greater Than 25";
  select Make, Model, MPG_City, MPG_Highway
    from sashelp.cars
    where MPG_City >25;
run;
quit;
proc sql;
title "Cars with City MPG Greater Than 40";
  select Make, Model, MPG_City, MPG_Highway
    from sashelp.cars
    where MPG_Highway >40;
run;
quit;
ods document close;
title;
proc document name=SqlTempl;
  list / levels=all;
run;
quit;
```
Program Description

In order to create a customized version of the output object’s table template with PROC DOCUMENT, you need to know the table template name. It is also helpful to see the SAS code that creates the default table template, so that you can customize the attributes.

Create the ODS document and the procedure output. The ODS DOCUMENT statement creates the document named SqlTempl. The SqlTempl document contains references to the PROC SQL output objects that the program creates.

```sas
ods document name=SqlTempl(write);
proc sql;
  title "Cars with City MPG Greater Than 25";
  select Make, Model, MPG_City, MPG_Highway
    from sashelp.cars
    where MPG_City >25;
run;
quit;
proc sql;
  title "Cars with City MPG Greater Than 40";
  select Make, Model, MPG_City, MPG_Highway
    from sashelp.cars
    where MPG_Highway >40;
run;
quit;
ods document close;
```

List the output objects. The PROC DOCUMENT procedure step that includes the LIST statement prints a listing of the paths for all of the output objects.

```sas
title;
proc document name=SqlTempl;
  list / levels=all;
run;
quit;
```

Output 4.3 Listing of the SqlTempl Document.

<table>
<thead>
<tr>
<th>Obs</th>
<th>Path</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>SQL#1</td>
<td>Dir</td>
</tr>
<tr>
<td>2</td>
<td>SQL#1\SQL_Results#1</td>
<td>Table</td>
</tr>
<tr>
<td>3</td>
<td>SQL#2</td>
<td>Dir</td>
</tr>
<tr>
<td>4</td>
<td>SQL#2\SQL_Results#1</td>
<td>Table</td>
</tr>
</tbody>
</table>
View the Table Templates

```bash
proc document name=SqlTempl;
  obtempl \SQL#1\SQL_Results#1;
  obtempl \SQL#2\SQL_Results#1;
run;
quit;
```

Program Description

**View the table template associated with the two tables.** By using the preceding LISTING statement, we know that the paths to the output objects that are tables are `\SQL#1\SQL_Results#1` and `\SQL#2\SQL_Results#1`. The OBTEMPL statement specifies that PROC DOCUMENT prints the contents of the output object’s table template to the open destination, which is the HTML destination in this example.

```bash
proc document name=SqlTempl;
  obtempl \SQL#1\SQL_Results#1;
  obtempl \SQL#2\SQL_Results#1;
run;
quit;
```

The name of the table template that is used for the PROC SQL output is Base.SQL. Both output objects use this table template because it is the default table template for all PROC SQL output. The entire code for the table template is written to the Results Viewer window.

**Output 4.4  SQL Table Template Program Code**

```bash
proc Template;
  define table Base.SQL;
  notes "PROC SQL Output";
  dynamic vmin flowmin flowmax ds;
  column obsno character flowcharacter numeric;
  define obsno;
      define header t;
       text _label_;
       def_split;
      end;
  header = t;
  space = 2;
  format = 8.0;
  just = r;
  style = Rowheader;
  id;
  generic;
end;
end;
```

Create Two Custom Table Templates

```bash
proc odstable name=Base.SQL store=mystore;
  cellstyle
    _DATANAME_ = "MPG_City" and _val_gt (40) as {background=orange},
    mod(_row_,2) as {Background=Honeydew};
run;
```

```bash
proc odstable name=Base.SQL;
  cellstyle
    _DATANAME_ = "MPG_Highway" and _val_gt (50) as {background=orange},
    mod(_row_,2) as {Background=Honeydew};
run;
```
Program Description

The following program creates two table templates, both named Base.SQL. The name Base.SQL is used because that is the table template name that PROC DOCUMENT is looking for. However, one instance of Base.SQL is stored in the default template store Sasuser.Templat, and the other is stored in the template store Work.Mystore. You can use the STORE= option in the REPLAY statement to replay two output objects and use a different table template for each one, even though the template name is the same.

Create a custom table template in a new item store. PROC ODSTABLE creates a custom table template. You must give the new template the same name as the default table template, Base.SQL. However, this template is now applied to every table created by SAS that uses the table name Base.SQL, unless it is overridden by another template created by PROC TEMPLATE or PROC ODSTABLE, removed with the DELETE statement, or manually removed from the item store. The STORE= option creates the item store Work.MyStore that contains the new table template.

```
proc odstable name=Base.SQL store=mystore;
  cellstyle
    _DATANAME_ = "MPG_City" and _val_ gt (40) as {background=orange},
    mod(_row_,2) as {Background=Honeydew};
run;
```

Create a second, different, custom table template in the Sasuser item store. Use PROC ODSTABLE to create a custom table template with the same name as the first table template. You must give the new template the same name as the default table template, Base.SQL. This template is applied to every table created by SAS that uses the table name Base.SQL, unless it is overridden by another template created by PROC TEMPLATE or PROC ODSTABLE, removed with the DELETE statement, or manually removed from the item store. However, because the STORE= option is not specified in the PROC ODSTABLE statement, the template is stored in the default location, which is Sasuser.Templat. Because these two new templates are stored in different locations, you can apply a different template to each output object.

```
proc odstable name=Base.SQL;
  cellstyle _DATANAME_ = "MPG_Highway" and _val_ gt (50) as {background=orange},
    mod(_row_,2) as {Background=Honeydew};
run;
```

Work.MyStore and Sasuser.Templat Item Stores

![Screenshot of SAS Environment showing templates in Work.MyStore and Sasuser.Templat Item Stores]

Replay the Output Objects

```
proc document name=sq1Temp1;
  replay \SQL#2\SQL_Results#1, \SQL#1\SQL_Results#1(store=mystore);
```
run;
quit;

Program Description

**Replay the output objects.** The REPLAY statement replays the output objects SQL_Results#1 and SQL_Results#2. The STORE= option specifies that the procedure use the template from the template store Mystore. If the STORE= option is not specified, then the Sasuser.Templat item store is used by default.

```
proc document name=sqlTemp;
    replay \SQL#2\SQL_Results#1, \SQL#1\SQL_Results#1(store=mystore);
run;
quit;
```

**Output 4.5  Output Using the Table Template from Work.Mystore**

<table>
<thead>
<tr>
<th>Make</th>
<th>Model</th>
<th>MPG (City)</th>
<th>MPG (Highway)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chevrolet</td>
<td>Aveo 4dr</td>
<td>28.000000</td>
<td>34.000000</td>
</tr>
<tr>
<td>Chevrolet</td>
<td>Aveo LS 4dr hatch</td>
<td>28.000000</td>
<td>34.000000</td>
</tr>
<tr>
<td>Chevrolet</td>
<td>Cavalier 2dr</td>
<td>26.000000</td>
<td>37.000000</td>
</tr>
<tr>
<td>Chevrolet</td>
<td>Cavalier 4dr</td>
<td>26.000000</td>
<td>37.000000</td>
</tr>
<tr>
<td>Chevrolet</td>
<td>Cavalier LS 2dr</td>
<td>26.000000</td>
<td>37.000000</td>
</tr>
<tr>
<td>Dodge</td>
<td>Neon SE 4dr</td>
<td>29.000000</td>
<td>36.000000</td>
</tr>
<tr>
<td>Dodge</td>
<td>Neon SXT 4dr</td>
<td>29.000000</td>
<td>36.000000</td>
</tr>
<tr>
<td>Ford</td>
<td>Focus ZX3 2dr hatch</td>
<td>26.000000</td>
<td>33.000000</td>
</tr>
<tr>
<td>Ford</td>
<td>Focus LX 4dr</td>
<td>27.000000</td>
<td>36.000000</td>
</tr>
<tr>
<td>Ford</td>
<td>Focus SE 4dr</td>
<td>26.000000</td>
<td>33.000000</td>
</tr>
<tr>
<td>Ford</td>
<td>Focus ZX5 5dr</td>
<td>26.000000</td>
<td>33.000000</td>
</tr>
<tr>
<td>Ford</td>
<td>Focus ZTW</td>
<td>26.000000</td>
<td>33.000000</td>
</tr>
<tr>
<td>Honda</td>
<td>Civic Hybrid 4dr manual (gas/electric)</td>
<td>46.000000</td>
<td>51.000000</td>
</tr>
<tr>
<td>Honda</td>
<td>Insight 2dr (gas/electric)</td>
<td>60.000000</td>
<td>66.000000</td>
</tr>
<tr>
<td>Honda</td>
<td>Civic DX 2dr</td>
<td>32.000000</td>
<td>38.000000</td>
</tr>
<tr>
<td>Ford</td>
<td>Fusion</td>
<td>36.000000</td>
<td>41.000000</td>
</tr>
</tbody>
</table>
Output 4.6 Output Using the Table Template from Sasuser.Templat

Delete the table templates. The DELETE statements delete the template Base.SQL from the Sasuser.Templat template store and the template Base.SQL from the Work.Mystore template store. Because Base.SQL is the default table template for all PROC SQL output, if you do not delete the custom templates, all of your PROC SQL output will have the customized template applied to it.

```sas
proc template;
   delete Base.SQL;
   delete Base.SQL / store=mystore;
run;
```
Part 2

ODS and Cascading Style Sheets

Chapter 5
Introduction to Cascading Style Sheets .................................. 39

Chapter 6
Working with the ODS Document Object Model ......................... 45

Chapter 7
Using Style Selectors .......................................................... 49

Chapter 8
SAS Dynamic Property Values ............................................... 57

Chapter 9
Resources and Examples ....................................................... 61
Chapter 5
Introduction to Cascading Style Sheets

Overview

Cascading style sheets (CSS) is a style sheet language that you can use with ODS to control the look and formatting of ODS output. A cascading style sheet is an external file that contains label-value pairs that describe the visual aspects of output, such as fonts, colors, borders, and so on. You can then apply the CSS to your ODS output. The CSS language that you can use with ODS is based on the standard CSS syntax that can be found on the Internet at http://www.w3.org/Style/CSS/. However, with ODS, you can apply CSS to all third-party formatted destinations except PowerPoint.

You can use CSS as an alternative to PROC TEMPLATE styles. Unlike PROC TEMPLATE styles, CSS enables you to apply custom styles based on column names, BY group names, BY group variables, and anchor names. Using CSS with ODS gives you greater flexibility in customizing your ODS reports.

Using CSS with ODS enables you to use the power of CSS style selectors with the internal data querying capabilities of PROC TEMPLATE styles. You can create custom output by defining your own external cascading style sheet file to apply to your ODS output. To use CSS with ODS, it is recommended that you perform the following steps:

1. Look at the ODS document object model (ODS DOM) to determine what elements and attributes exist so that you can construct your CSS selectors to address those areas. You can view the ODS DOM with the ODS TRACE DOM statement or with the DOM option on almost any ODS destination statement.

2. Create an external CSS file using standard CSS syntax.

3. Apply the CSS file to your ODS output with the CSSSTYLE= option in almost any ODS destination statement. The style sheet is interpreted by SAS instead of by the web browser.

In the following example, the style sheet "MyCss.css" is applied to PDF, RTF, and HTML output.

```sas
ods pdf cssstyle='MyCss.css';
ods rtf cssstyle='MyCss.css';
ods html cssstyle='MyCss.css';
proc print data=sashelp.class(obs=3);
run;
ods _all_ close;

**Figure 5.1** Default HTML Output

<table>
<thead>
<tr>
<th>Actual Product Sales</th>
<th>Product type</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>FURNITURE</td>
<td>OFFICE</td>
</tr>
<tr>
<td></td>
<td>Actual Sales</td>
<td>Actual Sales</td>
</tr>
<tr>
<td>Region by Division and Type</td>
<td>Sum</td>
<td>Sum</td>
</tr>
<tr>
<td>----------------------</td>
<td>--------------</td>
<td>--------------</td>
</tr>
<tr>
<td>Region</td>
<td>Division</td>
<td></td>
</tr>
<tr>
<td>EAST</td>
<td>CONSUMER</td>
<td>$72,570</td>
</tr>
<tr>
<td></td>
<td>EDUCATION</td>
<td>$73,901</td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td>$146,471</td>
</tr>
<tr>
<td>WEST</td>
<td>Division</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CONSUMER</td>
<td>$76,209</td>
</tr>
<tr>
<td></td>
<td>EDUCATION</td>
<td>$67,945</td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td>$144,154</td>
</tr>
<tr>
<td>Total</td>
<td>Division</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CONSUMER</td>
<td>$148,779</td>
</tr>
<tr>
<td></td>
<td>EDUCATION</td>
<td>$141,846</td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td>$290,625</td>
</tr>
</tbody>
</table>
Before you start working with cascading style sheets, you should be familiar with the following terminology.

cascading style sheet (CSS)

is a set of specifications that control the look and formatting of ODS output. You can use a CSS file to specify fonts, sizes, and colors of the text in ODS reports. A CSS file is external to SAS and contains a collection of rule sets, style properties, declaration blocks, and CSS selectors.

The easiest way to apply a CSS to your ODS report is by using the CSSSTYLE= option on almost any ODS destination statement. The CSSSTYLE= option is valid for all ODS destination statements except ODS LISTING, ODS OUTPUT, and ODS POWERPOINT. For information about ODS destination statements, see “Dictionary of ODS Language Statements” in *SAS Output Delivery System: User's Guide*.

You can also use the STYLESHEET= option in the ODS HTML statement or the IMPORT statement in PROC TEMPLATE to apply a CSS to your output. For information about the ODS HTML statement, see “ODS HTML Statement ” in *SAS Output Delivery System: User's Guide*. For information about the IMPORT statement, see “IMPORT Statement” in *SAS Output Delivery System: Procedures Guide*.
combinator
is an operator used between selectors to create more complex selectors that depend
on sibling-descendant relationships in the ODS document object model.

CSS rule set
is a combination of one or more CSS style selectors and one CSS declaration block.
There are usually multiple rule sets within a CSS file. The following code is an
eexample of one rule set that consists of one style selector named .SYSTEMTITLE
and one declaration block. The declaration block contains three style properties.

```
.systemtitle
{
  font-family: arial, helvetica, sans-serif;
  color: red;
  border: 1px solid black;
}
```

The following code is an example of one rule set that consists of five style selectors
that are separated by commas. There is one declaration block that contains one style
property.

```
.body, .systemtitle, .header, .rowheader, .data
{
  font-family: arial, sans-serif;
}
```

ODS document object model (DOM)
is an in-memory representation of the ODS report structure. Use the ODS TRACE
DOM statement or the DOM option to view the DOM for any ODS report. The
DOM option is valid for all ODS destination statements except ODS LISTING and
ODS OUTPUT.

DOM element
is a component of an ODS report that has been parsed into the ODS document object
model. Elements are written with a start element and an end element, with the
content in between. For example, a paragraph is represented by the <P> element, and
would be written as the following:

```
<p>This is a paragraph.</p>
```

style selector
is one or more character strings that identifies the part of the output that the
declaration block affects. You can specify one or more style selectors for each
declaration block. Multiple style selectors are separated by commas. There are
different types of style selectors. For detailed information about style selectors, see
Chapter 7, “Using Style Selectors,” on page 49.

In the rule set below, there is one style selector named .SYSTEMTITLE and one
declaration block that contains three style properties.

```
.systemtitle
{
  font-family: arial, helvetica, sans-serif;
  color: red;
  border: 1px solid black;
}
```

The following code is an example of one rule set that consists of five style selectors
that are separated by commas. There is one declaration block that contains one style
property.

```
.body, .systemtitle, .header, .rowheader, .data
```
CSS declaration block

is one or more style properties, enclosed in braces, that affect specific areas of ODS output. The following code is an example of one declaration block that contains three style properties.

```
{
    font-family: arial, helvetica, sans-serif;
    color: red;
    border: 1px solid black;
}
```

style property

is a visual property, such as color, font properties, and line characteristics, that is defined in a cascading style sheet file with a reserved name and value. Style properties are collectively referenced by a declaration block within a cascading style sheet. Style properties are separated by semi-colons.

In the following code, FONT-FAMILY, COLOR, and BORDER are style properties.

```
font-family: arial, helvetica, sans-serif;
color: red;
border: 1px solid black;
```
Chapter 6
Working with the ODS Document Object Model

ODS Document Object Model (DOM) Structure

The ODS document object model (DOM) is the in-memory representation of an ODS report. You can use the DOM to dynamically access and update the content, structure, and style of ODS output. By viewing the DOM, you can determine what elements and attributes exist so that you can construct your CSS selectors to access those areas. To view the DOM for a destination, use the DOM option in any ODS destination statement except ODS LISTING and ODS OUTPUT. You can also specify the DOM option in the ODS TRACE statement.

For example, the ODS TRACE DOM statement in the following code writes the trace record to the SAS log. The trace record includes the DOM for the PDF output.

```sas
ods html close;
ods trace dom;
ods pdf file="myFile.PDF";
proc report data=sashelp.class(obs=1);
  column name;
run;
ods pdf close;
ods trace off;
```

You can also use the DOM option in the ODS destination statement to write the DOM to the SAS log or to an external file.

```sas
ods html close;
ods pdf file="myFile.PDF" dom;
proc report data=sashelp.class(obs=1);
  column name;
run;
ods pdf close;
```
Log 6.1  SAS Log Output

199  ods html close;
200  ods pdf file="myFile.PDF" dom;
<!DOCTYPE html>
<html>
<head>
    <title>ODS PDF DOM</title>
    <link rel="stylesheet" href="style.css">
</head>
<body dest="pdf" class="body">
    proc report data=sashelp.class(obs=1);
    column name;
    run;

    <section id="idx" class="oo" data-name="procreporttable"
        label="detailed and/or summarized report" proc="report" output="report">
        <h1 class="systemtitle">
        </h1>
        <p class="systemtitle">Comparison of City and Highway Miles Per Gallon</p>
        <table class="table">
            <colgroup>
                <col type="char">
            </colgroup>
            <thead>
                <tr>
                    <th class="header" index="1">Name</th>
                </tr>
            </thead>
            <tbody>
                <tr>
                    <td class="data" type="char" unformatted-type="char" index="1"
                        name="_c1_
                        data-names="_c1_">Alfred</td>
                </tr>
            </tbody>
        </table>
    </section>

    ... more log output ...

    NOTE: There were 1 observations read from the data set SASHELP.CLASS.
    NOTE: PROCEDURE REPORT used (Total process time):
    real time           0.90 seconds
    cpu time            0.28 seconds

204  ods pdf close;
    <toc class="contentprocname">
    </div>
    </body>
</html>

1  The type of DOM is written in the <TITLE> elements at the beginning of the trace record (in this case, "ODS PDF DOM").

2  <SECTION id="idx" class="oo" data-name="procreporttable"
    label="detailed and/or summarized report" proc="report" output="report"> is the entire DOM element. SECTION is the element name.
id="idx" is an id= attribute and value pair.

class="oo" is a class= attribute and value pair.

The following are standard DOM element attributes and value pairs:

- data-name="procreporttable"
- label="detailed and/or summarized report"
- proc="report"
- output="report"

**Tips and Things to Remember**

The following tips can help you when working with ODS DOMs.

- It is recommended that you open one destination at a time when working with an ODS DOM. All of the trace information is written to the log, so if you have multiple destinations open, their DOMs are intermingled. This makes it harder to know what elements of the DOM correspond with specific destinations.

- You can use the DOM option on any ODS destination statement except ODS LISTING and ODS OUTPUT.

- If you use the ODS TRACE DOM statement, then use the ODS TRACE OFF statement after you close the ODS destination. This prevents your SAS log from filling up with trace information that you do not need.

- Each destination might have different DOMs. You must account for the differences in your CSS file.

- The DOM for all destinations resembles HTML markup so that standard CSS tools can be used to help analyze and apply styles to any destination.

- The type of DOM created is written in the title section of the DOM at the beginning of the trace record.
Chapter 7
Using Style Selectors

Overview

Style selectors identify the area of your report that you want to customize. Style selectors are similar to WHERE expressions in that they conditionally match a part of your ODS DOM. There are many ways to use style selectors. You can specify a simple style selector that matches an element name, or a complex selector that matches a pattern of inheritance between elements. Before you create your style selectors, you should be familiar with the ODS document object model (DOM) and the parts of a DOM element. For complete documentation about the ODS DOM, see Chapter 6, “Working with the ODS Document Object Model,” on page 45.

For industry-standard information about style selectors, see CSS Selectors.

The following is a typical DOM element.

```html
<section id="idx" class="oo" data-name="procreporttable"
label="detailed and/or summarized report" proc="report" output="report"> 
```

The DOM element has these parts:

1. `<SECTION id="idx" class="oo" data-name="procreporttable"
label="detailed and/or summarized report" proc="report" output="report">` is the entire DOM element. `SECTION` is the element name.
2 \text{id}="idx" \text{is an id= attribute and value pair.}
3 \text{class}="oo" \text{is a class= attribute and value pair.}
4 The following are standard DOM element attribute and value pairs:
   • \text{data-name}="procreporttable"
   • \text{label}="detailed and/or summarized report"
   • \text{proc}="report"
   • \text{output}="report"

\textbf{Class Selectors}

Class selectors are style selectors that select elements based on the value of the \text{class=} attribute in the markup of an ODS report. Class selectors must have a period (.) preceding the class name. For example, in the following rule set, the class style selector is \text{.SYSTEMTITLE}.

\begin{verbatim}
  .SYSTEMTITLE
  .systemtitle
  {
    font-family: arial, helvetica, sans-serif;
    color: red;
    border: 1px solid black;
  }
\end{verbatim}

The class selector \text{.SYSTEMTITLE} applies to the following markup elements:

\begin{verbatim}
<p class="systemtitle">The SAS System</p>
<p class="focused systemtitle large">The SAS System</p>
\end{verbatim}

However, the class selector \text{.SYSTEMTITLE} does not apply to the following markup elements:

\begin{verbatim}
<p>The SAS System</p>
<p class="footer">The SAS System</p>
<p class="system title">The SAS System</p>
\end{verbatim}

\textbf{Element Selectors}

Element selectors are style selectors that select DOM elements based on the element name. For example, the following rule set selects elements with the name \text{p}:

\begin{verbatim}
  p {color: green}
\end{verbatim}

All of the following HTML markup examples use the \text{<P>} element. The declaration block \text{\{color: green\}} would be applied to them.

\begin{verbatim}
<p class="systemtitle">The SAS System</p>
<p class="focused systemtitle large">The SAS System</p>
<p>The SAS System</p>
<p class="footer">The SAS System</p>
<p class="system title">The SAS System</p>
\end{verbatim}
Universal Selector

The universal selector is a style selector that is a wildcard. It can match any element name. The syntax for the universal selector is an asterisk (\*).

Pseudo-Class Selectors

Pseudo-class selectors are style selectors that select elements based on the relationships between DOM elements. Pseudo-classes are represented by the pseudo-class name prefixed with a colon (:). The following are examples of some ways that you can use pseudo-class selectors.

- select the first and last child of a parent element
- select a specific child based on its positional index in the parent element
- select an element by position of a particular element name

<table>
<thead>
<tr>
<th>Pseudo-class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>:root</td>
<td>Selects the top-level element in the DOM.</td>
</tr>
<tr>
<td>:first-child</td>
<td>Selects the first element within the parent.</td>
</tr>
<tr>
<td>:first-of-type</td>
<td>Selects the first element of that type (that is, same element name) in the parent.</td>
</tr>
<tr>
<td>:marker</td>
<td>Selects a list item bullet in printer output.</td>
</tr>
<tr>
<td>:nth-child(an+b)</td>
<td>Selects an element based on the equation an + b. This equation selects every ath element starting with element at position b. The equation can be replaced with the keywords even or odd for the simple case of alternating the selection.</td>
</tr>
<tr>
<td>:nth-of-type(an+b)</td>
<td>Selects the :nth-child, except that only the same elements of the same type are used in the calculation.</td>
</tr>
<tr>
<td>:empty</td>
<td>Selects one or more empty elements. This only applies to elements that have been specified as empty by the procedure.</td>
</tr>
<tr>
<td>::before</td>
<td>Inserts content before the element.</td>
</tr>
<tr>
<td>::after</td>
<td>Inserts content after the element.</td>
</tr>
</tbody>
</table>
Pseudo-class | Description
--- | ---
:not(...) | Selects an element if the selector within the argument is not true.

* This is a special case that is used the same way as the PRETEXT, PREIMAGE, POSTTEXT, and POSTIMAGE style attributes.

For an example that uses pseudo-class selectors to change the style of table cells, see “Example 3: Applying Styles to Table Cells” on page 88. For an example that uses pseudo-class selectors to change the style of lists, see “Example 2: Applying Styles to Lists” on page 80.

---

**ID Selectors**

ID selectors are style selectors that select elements based on the **id** attribute of a DOM element. The ID must be unique within a DOM and only one can be specified in the **id** attribute. ID selectors are indicated by a "#" prefix. The following is a CSS rule set with an ID selector:

```css
#idx1 { font-style: italic }
```

The rule set matches the following element:

```html
<section id="idx1"><p>The SAS System</p></section>
```

---

**Attribute Selectors**

Attribute selectors select DOM elements with the specified attribute. ID selectors and class selectors are special case attribute selectors. Attribute selectors use the following syntax to select attributes:

```
[attribute operator "value"]
```

* **operator** specifies the operator.

  * **operator** allows partial matches.

  * **Table 7.2**  Operators That Can Be Used with Attribute Selectors

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>=</td>
<td>Matches the entire attribute value.</td>
</tr>
<tr>
<td>^=</td>
<td>Matches the beginning of an attribute value.</td>
</tr>
<tr>
<td>=$</td>
<td>Matches the end of an attribute value.</td>
</tr>
<tr>
<td>*=</td>
<td>Matches any substring in an attribute value.</td>
</tr>
<tr>
<td>Operator</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
</tr>
<tr>
<td><code>~~</code></td>
<td>Matches any space-separated word in an attribute value. This operator can be used to emulate the class selector.</td>
</tr>
<tr>
<td>`</td>
<td>=</td>
</tr>
</tbody>
</table>

## Combinators

Combinators are characters that select an element based partially on its context within another element. This is done by combining selectors using one of the following characters:

- ` " " ` (space)
  indicates that the selector to the left must match an element anywhere in the parentage of the currently selected element.

  **Example**  The following selector selects all elements with the class="header" attribute and an ancestor with the id="idx1" attribute.
  ```html
  #IDX1 .header
  ```

- `>`
  selects elements that are a direct descendent of the specified element.

  **Example**  Suppose that you have two nested tables where the outer `<TABLE>` element contains a class="table" attribute and the inner `<TABLE>` element contains a class="stacked" attribute.
  ```html
  <table cellspacing="2" cellpadding="2" border="0">
    <tr>
      <th></th>
      <th></th>
    </tr>
    <tr>
      <td></td>
      <td></td>
    </tr>
  </table>
  ```
  ```html
  <table cellspacing="2" cellpadding="2" border="0">
    <tr>
      <th></th>
      <th></th>
    </tr>
    <tr>
      <td></td>
      <td></td>
    </tr>
  </table>
  ```
You only want to customize the `<TH>` elements of the outer table. You can use the following direct descendant combinator to customize only the outer table `<TH>` elements:

```
.table > tr > th
```

You can use the following selector to customize only the inner table `<TH>` elements:

```
.table > * > tr > * > .stacked > tr > th
```

~ selects elements that have another sibling anywhere within the parent.

**Example**

To select any `<TD>` element that also has a `<TH>` element sibling anywhere within the DOM, you can use the following selector.

```
th ~ td
```

This selector selects both `<TD>` elements in the last row of the HTML table.

+ selects elements that have a specified element immediately preceding them.

**Example**

To select any `<TD>` element that also has a `<TH>` element sibling anywhere within the DOM, you can use the following selector.

```
th + td
```

This selector selects both `<TD>` elements in the last row of the HTML table.

For an example that uses combinators, see “Example 3: Applying Styles to Table Cells” on page 88.

---

### Using a Single Style Selector Type per Rule Set

You can specify a single style selector for each rule set. The following code contains two DOM elements taken from the same DOM. The first table element creates a container for the system titles. The second table element creates the table for the output.

```
<table class="systitleandfootercontainer">
  ...
</table>

<table class="table">
  ...
</table>
```

The following table shows some example of selectors, the complete rule set, and results.

<table>
<thead>
<tr>
<th>Selector Type</th>
<th>Selector</th>
<th>Rule Set</th>
</tr>
</thead>
<tbody>
<tr>
<td>element</td>
<td>table</td>
<td>table {background-color: yellow; }</td>
</tr>
<tr>
<td>class</td>
<td>.systitleandfootercontainer</td>
<td>.systitleandfootercontainer {background-color: yellow; }</td>
</tr>
</tbody>
</table>
Using Multiple Style Selector Types per Rule Set

You can specify multiple selector types for one rule set. The following code contains one DOM element, a `<SECTION>` element. Any of the selectors in the following table select this element.

```html
...more elements...

<section id="idx" class="oo" data-name="report" label="table 1" proc="tabulate"
  output="table" contents-label="table 1" data-label="table 1">
  ...more elements...
</section>
```

Table 7.4  Examples of Multiple Selectors

<table>
<thead>
<tr>
<th>Selector Type</th>
<th>Selector</th>
<th>Rule Set</th>
</tr>
</thead>
<tbody>
<tr>
<td>Element and class</td>
<td>section.oo</td>
<td>section.oo {background-color: yellow; }</td>
</tr>
<tr>
<td>Element, attribute, and class</td>
<td>section .oo[contents-label=&quot;table-1&quot;]</td>
<td>section[contents-label=&quot;table-1&quot;].oo {background-color: yellow; }</td>
</tr>
<tr>
<td>Element and class attribute</td>
<td>section[class=&quot;oo&quot;]</td>
<td>section[class=&quot;oo&quot;] {background-color: yellow; }</td>
</tr>
<tr>
<td>Class attribute and attribute</td>
<td>[class=&quot;oo&quot;][data-label=&quot;table-1&quot;]</td>
<td>[class=&quot;oo&quot;][data-label=&quot;table 1&quot;] {background-color: yellow; }</td>
</tr>
<tr>
<td>Class and partial ID attribute match</td>
<td>.oo[data-label~=&quot;table&quot;]</td>
<td>.oo[data-label~=&quot;table&quot;] {background-color: yellow; }</td>
</tr>
<tr>
<td>Universal selector (matches every element)</td>
<td>*</td>
<td>*{background-color: yellow; }</td>
</tr>
</tbody>
</table>

Tips for Working with Style Selectors

- Selector types can be combined to form more complex selectors.
- Style properties in later rule sets take precedence over those in earlier rule sets.
- It is recommended that you start simply, use a boilerplate, and add on from there.
• ID and class selectors are special cases of attribute selectors and can be specified with the attribute selector syntax. However, it is more efficient to use the dedicated syntax for ID and class selectors.

• ID and class selectors cannot do partial matches. If you want to select ID or class attributes based on partial matches, use the general attribute selector syntax.

• If you want all `<SECTION>` elements to have a specific style, you would use the element type selector. However, if you want to select a specific `<SECTION>` element that contains the `id="idx1"` attribute, then you could use the ID selector `id="idx1"`.

• When specifying multiple selector types, specify element type selectors first. There is no syntactical prefix for element selectors, so they should be specified first.

• Attribute selectors are not as efficient as other types of selectors. It is a good practice to prefix an attribute selector with an element name selector, class name selector, or ID selector.
The following dynamic functions can be specified as the value of any CSS style property.

**SYMGET(‘macro-variable’)***

specifies that the value of the *macro-variable* is used as the value of the style property.

*macro-variable*

is the name of a macro variable. The name can refer to a new or existing macro variable.

Example  “Example 4: Using Dynamic Property Values” on page 97

**RESOLVE(&macro-variable) | (%macro)**

specifies that the value of the &macro-variable is used as the value of the style property.

&macro-variable

is the name of a macro variable preceded by an ampersand (&). The name can refer to a new or existing macro variable.

%macro

is a character expression that produces a text expression for resolution by the SAS macro facility.

Example  “Example 4: Using Dynamic Property Values” on page 97

**FORMAT(SAS-format)**

specifies that a *SAS-format* is used as the value of the style property.

Example  “Example 4: Using Dynamic Property Values” on page 97

**EXPRESSION(‘expression’)**

specifies that the result of *expression* is used as the value of the style property. The EXPRESSION function in the following style property uses red and black font colors when the _DATANAME_ built-in variable is equal to ‘type’.

{
In the preceding code, the components of the expression are the following:

**CHOOSEC** is a SAS function that returns a character value that represents the results of choosing from a list of arguments.

_**DATANAME**_ is the data column name.

‘type’ is the value of _**DATANAME**_.

red, black are the font colors to use if the expression is true.

**expression** is a section of programming code that is evaluated for each item that contains a numeric variable.

**expression** has this form:

**expression-1 <comparison-operator expression-n>**

**expression** is an arithmetic or logical programming code that consists of a sequence of operators and operands. An operator is a symbol that requests a comparison, logical operation, or arithmetic calculation. An operand is one of the following:

**constant** is a fixed value such as the name of a column or symbols that are declared in a DYNAMIC, MVAR, or NMVAR statement in the current template.

**SAS function** specifies a SAS function. For information about SAS functions, see *SAS Functions and CALL Routines: Reference*.

**comparison-operator** compares a variable with a value or with another variable. The following table lists the comparison operators:

**Table 8.1 Comparison Operators**

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Mnemonic Equivalent</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>=</td>
<td>EQ</td>
<td>Equal to</td>
</tr>
<tr>
<td>^= or &lt;= or %= or &lt;&gt;</td>
<td>NE</td>
<td>Not equal to</td>
</tr>
<tr>
<td>&gt;</td>
<td>GT</td>
<td>Greater than</td>
</tr>
<tr>
<td>&lt;</td>
<td>LT</td>
<td>Less than</td>
</tr>
<tr>
<td>&gt;=</td>
<td>GE</td>
<td>Greater than or equal to</td>
</tr>
<tr>
<td>&lt;=</td>
<td>LE</td>
<td>Less than or equal to</td>
</tr>
<tr>
<td></td>
<td>IN</td>
<td>Equal to one or more from a list of values</td>
</tr>
</tbody>
</table>
See You can use any expression that can be used in the WHERE= data set option. For information about expressions that you can use in the WHERE= data set option, see the WHERE= data set option in SAS Data Set Options: Reference and the section on WHERE-Expression Processing in SAS Language Reference: Concepts.
Common Industry-Standard Style Properties

The following style properties are commonly used in CSS. However, a complete list of style properties can be found at W3C Cascading Style Sheets Home Page.

**BACKGROUND:** `<color> <background-attachment> <background-position> <background-repeat> <URL("url")>`

specifies background attributes.

The BACKGROUND: property in the following code specifies the following background attributes:

- red background color
- the background image "Sas.gif"
- the image is not repeated
- the image is fixed
- the image is centered

```css
body {
  background: red url('sas.gif') no-repeat fixed center;
}
```

color

is a keyword or numerical value that specifies a color. For industry-standard documentation, see [http://www.w3.org/wiki/CSS/Properties/color](http://www.w3.org/wiki/CSS/Properties/color) or [http://www.w3.org/TR/CSS2/syndata.html#value-def-color](http://www.w3.org/TR/CSS2/syndata.html#value-def-color). You can represent a color in one of the following formats:
hexadecimal
  is a six-digit, three-byte hexadecimal number used to represent colors.

RGB
  is a red/green/blue value.

RGBA
  is a red/green/blue/alpha value.

HSL
  is a hue/saturation/lightness value.

HSLA
  is a hue/saturation/light/alpha value.

**background-attachment**
  specifies whether a background image is fixed or scrolls with the rest of the page.
  **background-attachment** can be one of the following:

  SCROLL
    specifies that the background image scrolls with the page.

  FIXED
    specifies that the background image does not scroll with the page.

  Default: SCROLL

**background-position**
  specifies the position of the background images. **background-position** can be one of the following:

  **keyword**
    specifies a keyword as the position of the background image. **keyword** can be one of the following:

    • CENTER
    • LEFT
    • RIGHT
    • TOP
    • BOTTOM

  **x% y%**
    specifies percent (%) values for the horizontal and vertical position of the image.

  **x%**
    specifies the horizontal position followed by a % sign.

  **y%**
    specifies the vertical position followed by a % sign.

  **Tip** The top left corner is 0% 0%. The right bottom corner is 100% 100%. If you specify only one value, the other value is 50%.

  **x-unit y-unit**
    specifies fixed values for the horizontal and vertical position of the image.

  **x-unit**
    specifies the horizontal position followed by a unit of measure.

  **y-unit**
    specifies the vertical position followed by a unit of measure.
Table 9.1 Units of Measure for Dimension

<table>
<thead>
<tr>
<th>Unit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cm</td>
<td>Centimeters</td>
</tr>
<tr>
<td>em</td>
<td>Standard typesetting measurement unit for width</td>
</tr>
<tr>
<td>ex</td>
<td>Standard typesetting measurement unit for height</td>
</tr>
<tr>
<td>in</td>
<td>Inches</td>
</tr>
<tr>
<td>mm</td>
<td>Millimeters</td>
</tr>
<tr>
<td>pc</td>
<td>Picas</td>
</tr>
<tr>
<td>pt</td>
<td>A printer's point</td>
</tr>
<tr>
<td>px</td>
<td>Pixels</td>
</tr>
</tbody>
</table>

Default 0% 0%

Tip The background image is placed according to the background-position property. If no background-position is specified, the image is always placed at the element's top left corner.

background-repeat specifies how to repeat the background images. background-repeat can be one of the following:

REPEAT specifies that the background image is repeated both vertically and horizontally.

REPEAT-X specifies that the background image is repeated horizontally.

REPEAT-Y specifies that the background image is repeated vertically.

NO-REPEAT specifies that the background-image is not repeated.

Default REPEAT

url is a reference to an external resource.

Requirement url must be enclosed in quotation marks and parenthesis.

BACKGROUND-COLOR: color specifies the background color.

color is a keyword or numerical value that specifies a color. For industry-standard documentation, see http://www.w3.org/wiki/CSS/Properties/color or http://www.w3.org/TR/CSS2/syndata.html#value-def-color. You can represent a color in one of the following formats:
hexadecimal
  is a six-digit, three-byte hexadecimal number used to represent colors.

RGB
  is a red/green/blue value.

RGBA
  is a red/green/blue/alpha value.

HSL
  is a hue/saturation/lightness value.

HSLA
  is a hue/saturation/light/alpha value.

**BACKGROUND-IMAGE: URL("url")**
specifies a background image.

*url* is a reference to an external resource.

**Requirement** *url* must be enclosed in quotation marks and parenthesis.

**BORDER-side-COLOR: color**
specifies the border color of the specified side.

*color*
  is a keyword or numerical value that specifies a color. For industry-standard
documentation, see http://www.w3.org/wiki/CSS/Properties/color or http://
www.w3.org/TR/CSS2/syndata.html#value-def-color. You can represent a color
in one of the following formats:

hexadecimal
  is a six-digit, three-byte hexadecimal number used to represent colors.

RGB
  is a red/green/blue value.

RGBA
  is a red/green/blue/alpha value.

HSL
  is a hue/saturation/lightness value.

HSLA
  is a hue/saturation/light/alpha value.

*side*
specifies the side of a cell or table.

*side* is one of the following:
  • TOP
  • BOTTOM
  • RIGHT
  • LEFT

**BORDER-COLOR: color-1 <, ..., color-4>**
specifies the color of one or more borders.

*color*
  is a keyword or numerical value that specifies a color. For industry-standard
documentation, see http://www.w3.org/wiki/CSS/Properties/color or http://
You can represent a color in one of the following formats:

- **hexadecimal**
  
  - is a six-digit, three-byte hexadecimal number used to represent colors.

- **RGB**
  
  - is a red/green/blue value.

- **RGBA**
  
  - is a red/green/blue/alpha value.

- **HSL**
  
  - is a hue/saturation/lightness value.

- **HSLA**
  
  - is a hue/saturation/light/alpha value.

The order in which you specify the colors determines the color applied to each side.

<table>
<thead>
<tr>
<th>Example</th>
<th>Order</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>{border-color:red;}</code></td>
<td>All four sides</td>
<td>Red</td>
</tr>
<tr>
<td><code>{border-color:red purple;}</code></td>
<td>Top and bottom</td>
<td>Red</td>
</tr>
<tr>
<td></td>
<td>Right and left</td>
<td>Purple</td>
</tr>
<tr>
<td><code>{border-color:red purple orange;}</code></td>
<td>Top</td>
<td>Red</td>
</tr>
<tr>
<td></td>
<td>Right and left</td>
<td>Purple</td>
</tr>
<tr>
<td></td>
<td>Bottom</td>
<td>Orange</td>
</tr>
<tr>
<td><code>{border-color:red purple orange yellow;}</code></td>
<td>Top</td>
<td>Red</td>
</tr>
<tr>
<td></td>
<td>Right</td>
<td>Purple</td>
</tr>
<tr>
<td></td>
<td>Bottom</td>
<td>Orange</td>
</tr>
<tr>
<td></td>
<td>Left</td>
<td>Yellow</td>
</tr>
</tbody>
</table>

**BORDER-side-STYLE: line-style**

specifies the border style of the specified side.

- **line-style**
  
  specifies the line style. *line-style* is one of the following:
  
  - DASHED
  - DOTTED
  - DOUBLE
  - DOTTED
  - GROOVE
  - RIDGE
  - HIDDEN
  - SOLID

- **side**
  
  specifies one side of a cell or table.
  
  *side* is one of the following:
  
  - TOP
  - BOTTOM
  - RIGHT
• LEFT

**BORDER-STYLE: line-style-1 <, ..., line-style-4>**
specifies the border style of one or more borders.

*line-style*
specifies the line style. *line-style* is one of the following:

- DASHED
- DOTTED
- DOUBLE
- GROOVE
- RIDGE
- HIDDEN
- SOLID

The order in which you specify the style determines the style applied to each side.

<table>
<thead>
<tr>
<th>Example</th>
<th>Order</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>{border-style: double;}</code></td>
<td>All four sides</td>
<td>Double</td>
</tr>
<tr>
<td><code>{border-style: solid double;}</code></td>
<td>Top and bottom</td>
<td>Solid</td>
</tr>
<tr>
<td></td>
<td>Right and left</td>
<td>Double</td>
</tr>
<tr>
<td><code>{border-style: none double solid;}</code></td>
<td>Top</td>
<td>None</td>
</tr>
<tr>
<td></td>
<td>Right and left</td>
<td>Double</td>
</tr>
<tr>
<td></td>
<td>Bottom</td>
<td>Solid</td>
</tr>
<tr>
<td><code>{border-style: none double solid none;}</code></td>
<td>Top</td>
<td>None</td>
</tr>
<tr>
<td></td>
<td>Right</td>
<td>Double</td>
</tr>
<tr>
<td></td>
<td>Bottom</td>
<td>Solid</td>
</tr>
<tr>
<td></td>
<td>Left</td>
<td>None</td>
</tr>
</tbody>
</table>

**BORDER-side-WIDTH: fixed-value**
specifies the border width of the specified side.

*side* is one of the following

- TOP
- BOTTOM
- RIGHT
- LEFT

*fixed-value*
is a number followed by one of the following units of measure:

<table>
<thead>
<tr>
<th>cm</th>
<th>Centimeters</th>
</tr>
</thead>
<tbody>
<tr>
<td>em</td>
<td>Standard typesetting measurement unit for width</td>
</tr>
<tr>
<td>ex</td>
<td>Standard typesetting measurement unit for height</td>
</tr>
<tr>
<td>in</td>
<td>Inches</td>
</tr>
<tr>
<td>mm</td>
<td>Millimeters</td>
</tr>
<tr>
<td>----</td>
<td>-------------</td>
</tr>
<tr>
<td>pc</td>
<td>Picas</td>
</tr>
<tr>
<td>pt</td>
<td>A printer's point</td>
</tr>
<tr>
<td>px</td>
<td>Pixels</td>
</tr>
</tbody>
</table>

**Restriction**  
`fixed-value` must be followed by a unit of measure.

**BORDER-WIDTH:** `fixed-value-1 <, ..., fixed-value-4>` specifies the border width of one or more borders.

`fixed-value` is a number followed by one of the following units of measure:

<table>
<thead>
<tr>
<th>cm</th>
<th>Centimeters</th>
</tr>
</thead>
<tbody>
<tr>
<td>em</td>
<td>Standard typesetting measurement unit for width</td>
</tr>
<tr>
<td>ex</td>
<td>Standard typesetting measurement unit for height</td>
</tr>
<tr>
<td>in</td>
<td>Inches</td>
</tr>
<tr>
<td>mm</td>
<td>Millimeters</td>
</tr>
<tr>
<td>pc</td>
<td>Picas</td>
</tr>
<tr>
<td>pt</td>
<td>A printer's point</td>
</tr>
<tr>
<td>px</td>
<td>Pixels</td>
</tr>
</tbody>
</table>

**Restriction**  
`fixed-value` must be followed by a unit of measure.

The order in which you specify the widths determines the width applied to each side.

<table>
<thead>
<tr>
<th>Example</th>
<th>Order</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>{border-width:3mm;}</code></td>
<td>All four sides</td>
<td>3mm</td>
</tr>
<tr>
<td><code>{border-width:3mm 6mm;}</code></td>
<td>Top and bottom</td>
<td>3mm</td>
</tr>
<tr>
<td></td>
<td>Right and left</td>
<td>6mm</td>
</tr>
<tr>
<td><code>{border-width:3mm 6mm 9mm;}</code></td>
<td>Top</td>
<td>3mm</td>
</tr>
<tr>
<td></td>
<td>Right and left</td>
<td>6mm</td>
</tr>
<tr>
<td></td>
<td>Bottom</td>
<td>9mm</td>
</tr>
</tbody>
</table>
**Example** | **Order** | **Result**
---|---|---
{border-width:3mm 6mm 9mm 12mm;} | Top | 3mm
| Right | 6mm
| Bottom | 9mm
| Left | 12mm

**BORDER-SPACING:** *fixed-value*
specifies the length between table cells.

*fixed-value*
is a number followed by one of the following units of measure:

<table>
<thead>
<tr>
<th>unit</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cm</td>
<td>Centimeters</td>
</tr>
<tr>
<td>em</td>
<td>Standard typesetting measurement unit for width</td>
</tr>
<tr>
<td>ex</td>
<td>Standard typesetting measurement unit for height</td>
</tr>
<tr>
<td>in</td>
<td>Inches</td>
</tr>
<tr>
<td>mm</td>
<td>Millimeters</td>
</tr>
<tr>
<td>pc</td>
<td>Picas</td>
</tr>
<tr>
<td>pt</td>
<td>A printer’s point</td>
</tr>
<tr>
<td>px</td>
<td>Pixels</td>
</tr>
</tbody>
</table>

**Restriction** *fixed-value* must be followed by a unit of measure.

**COLOR:** *color*
specifies the font color.

*color*
is a keyword or numerical value that specifies a color. For industry-standard documentation, see [http://www.w3.org/wiki/CSS/Properties/color](http://www.w3.org/wiki/CSS/Properties/color) or [http://www.w3.org/TR/CSS2/syndata.html#value-def-color](http://www.w3.org/TR/CSS2/syndata.html#value-def-color). You can represent a color in one of the following formats:

- **hexadecimal**
  - is a six-digit, three-byte hexadecimal number used to represent colors.

  - **RGB**
    - is a red/green/blue value.

  - **RGBA**
    - is a red/green/blue/alpha value.

- **HSL**
  - is a hue/saturation/lightness value.

- **HSLA**
  - is a hue/saturation/light/alpha value.
FONT-FAMILY: "font-1", "font-2", ...

specifies one or more fonts to use in an element. If you supply multiple fonts, then the destination device uses the first one that is installed on the system.

font

specifies the name of the font.

ODS can use TrueType fonts. All Universal Printers and many SAS/GRAPH devices use the FreeType library to render TrueType fonts for output in all of the operating environments that SAS software supports. In addition to SAS Monospace and SAS Monospace Bold, 21 new TrueType fonts are made available when you install SAS:

• five Latin fonts compatible with Microsoft
• eight multilingual Unicode fonts
• eight monolingual Asian fonts

For more information about the TrueType fonts, see “Using Fonts with Universal Printers and SAS/GRAPH Devices” in SAS Language Reference: Concepts.

Requirements

You must separate multiple fonts with commas.

You must enclose font in quotation marks if the font name contains a space, such as "Times New Roman".

Tip

If you specify more than one font, then the destination device uses the first one that is installed on the system.

FONT-SIZE: fixed-value | percentage

specifies the font size as a fixed value or percent.

fixed-value

is a number followed by one of the following units of measure:

<table>
<thead>
<tr>
<th>cm</th>
<th>Centimeters</th>
</tr>
</thead>
<tbody>
<tr>
<td>em</td>
<td>Standard typesetting measurement unit for width</td>
</tr>
<tr>
<td>ex</td>
<td>Standard typesetting measurement unit for height</td>
</tr>
<tr>
<td>in</td>
<td>Inches</td>
</tr>
<tr>
<td>mm</td>
<td>Millimeters</td>
</tr>
<tr>
<td>pc</td>
<td>Picas</td>
</tr>
<tr>
<td>pt</td>
<td>A printer's point</td>
</tr>
<tr>
<td>px</td>
<td>Pixels</td>
</tr>
</tbody>
</table>

Restriction

fixed-value must be followed by a unit of measure.

percentage

specifies the percentage of the containing element value followed by a percent (%) sign.
Restriction  percentage must be followed by a percent sign (%).

**FONT-STYLE: ITALIC | OBLIQUE | NORMAL**
specifies the font style for an element.

**FONT-WEIGHT: BOLD | NORMAL**
specifies the thickness of the character outlines relative to their height.

**FONT:** "font-style"> "font-variant"> "font-weight"> "font-size" > "font-family"
specifies a list of font attributes.

*font-style*
specifies the font style for an element. *font-style* can be one of the following:

- ITALIC
- OBLIQUE
- NORMAL

**Requirement**  *font-style* must be enclosed in quotation marks.

*font-variant*
specifies whether the text is displayed in a small caps font. *font-variant* can be one of the following:

- NORMAL
  specifies the text is displayed in the normal font.

- SMALL-CAPS
  specifies that the text is displayed in a small caps font. A small caps font is a font where all lowercase letters have been converted to uppercase letters.

**Default**  NORMAL

**Requirement**  *font-variant* must be enclosed in quotation marks.

*font-weight*
specifies the thickness of the character outlines relative to their height. *font-weight* can be one of the following:

- BOLD
- NORMAL

**Requirement**  *font-weight* must be enclosed in quotation marks.

*font-size*
specifies the font size as a fixed value or percentage.

**Requirement**  *font-size* must be enclosed in quotation marks.

**See**  “FONT-SIZE: fixed-value | percentage ” on page 69

*line-height*
specifies the minimal height of line boxes within the element.

**Requirement**  *line-height* must be enclosed in quotation marks.

*font-family*
specifies a font name.
Requirement  *font-family* must be enclosed in quotation marks if the font family name contains a space.

See  “FONT-FAMILY: "font-1", "font-2", ...” on page 69

Restriction  Font size, line height, and font family must be specified in the following order: *font-size*, *line-height*, *font-family*.

PADDING: *padding-1* <, ..., *padding-4*>

specifies the amount of white space between the content of the element and the border. You can specify up to four values. *padding* can be specified as a fixed value or a percentage.

*fixed-value*

is a number followed by one of the following units of measure:

<table>
<thead>
<tr>
<th>Unit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cm</td>
<td>Centimeters</td>
</tr>
<tr>
<td>em</td>
<td>Standard typesetting measurement unit for width</td>
</tr>
<tr>
<td>ex</td>
<td>Standard typesetting measurement unit for height</td>
</tr>
<tr>
<td>in</td>
<td>Inches</td>
</tr>
<tr>
<td>mm</td>
<td>Millimeters</td>
</tr>
<tr>
<td>pc</td>
<td>Picas</td>
</tr>
<tr>
<td>pt</td>
<td>A printer's point</td>
</tr>
<tr>
<td>px</td>
<td>Pixels</td>
</tr>
</tbody>
</table>

Restriction  *fixed-value* must be followed by a unit of measure.

*percentage*

specifies the percentage of the containing element value followed by a percent (%) sign.

Restriction  *percentage* must be followed by a percent sign (%).

The order in which you specify the values determines the padding applied to each side.

<table>
<thead>
<tr>
<th>Example</th>
<th>Order</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>{padding:20px;}</td>
<td>All four sides</td>
<td>20px</td>
</tr>
<tr>
<td>{padding:5px 10px;}</td>
<td>Top and bottom, Right and left</td>
<td>5px 10px</td>
</tr>
<tr>
<td>Example</td>
<td>Order</td>
<td>Result</td>
</tr>
<tr>
<td>-------------------------</td>
<td>------------------------</td>
<td>--------</td>
</tr>
<tr>
<td>{padding:5px 10px 20px;}</td>
<td>Top 5px</td>
<td>5px</td>
</tr>
<tr>
<td></td>
<td>Right and left 10px</td>
<td>10px</td>
</tr>
<tr>
<td></td>
<td>Bottom 20px</td>
<td>20px</td>
</tr>
<tr>
<td>{padding:5px 10px 15px 20px;}</td>
<td>Top 5px</td>
<td>5px</td>
</tr>
<tr>
<td></td>
<td>Right 10px</td>
<td>10px</td>
</tr>
<tr>
<td></td>
<td>Bottom 15px</td>
<td>15px</td>
</tr>
<tr>
<td></td>
<td>Left 20px</td>
<td>20px</td>
</tr>
</tbody>
</table>

**PADDING-side: fixed-value | percentage**

Specifies the amount of white space between the content of the element and the border.

- **side**
  - Specifies one side of a table or cell.
  - **side** is one of the following:
    - TOP
    - BOTTOM
    - RIGHT
    - LEFT

- **fixed-value**
  - Is a number followed by one of the following units of measure:

<table>
<thead>
<tr>
<th>Unit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cm</td>
<td>Centimeters</td>
</tr>
<tr>
<td>em</td>
<td>Standard typesetting measurement unit for width</td>
</tr>
<tr>
<td>ex</td>
<td>Standard typesetting measurement unit for height</td>
</tr>
<tr>
<td>in</td>
<td>Inches</td>
</tr>
<tr>
<td>mm</td>
<td>Millimeters</td>
</tr>
<tr>
<td>pc</td>
<td>Picas</td>
</tr>
<tr>
<td>pt</td>
<td>A printer's point</td>
</tr>
<tr>
<td>px</td>
<td>Pixels</td>
</tr>
</tbody>
</table>

**Restriction**

- **fixed-value** must be followed by a unit of measure.

- **percentage**
  - Specifies the percentage of the containing element value followed by a percent (%) sign.
**Documentation Resources and Websites**

Using ODS with CSS:


Industry-standard CSS documentation:

- [W3C Cascading Style Sheets Home Page](http://www.w3.org/)
- [www.w3.org Information about colors](http://www.w3.org/)
- [www.w3.org More information about colors](http://www.w3.org/)
- [www.w3.org Information about style selectors](http://www.w3.org/)

ODS TRACE statement:

For complete information about the ODS TRACE statement, see “ODS TRACE Statement” in *SAS Output Delivery System: Procedures Guide*.

**Examples**

**Example 1: Using Colors and Class Selectors to Highlight Output Areas**

**Features:**
- CSSSTYLE= option
- DOM option
- ODS HTML CLOSE statement
- ODS PDF statement

**Other features:**
- Industry-standard CSS style selectors
  - Class selectors
  - Attribute selectors
- Industry-standard CSS style properties
  - BACKGROUND-COLOR:
  - COLOR:
  - FONT-WEIGHT:
Details
When using CSS styles and the DOM, it is helpful to use colors or patterns to help you see the area of your output that is affected by your style selectors. The following example uses colors and class selectors to highlight the areas of the output that each class selector corresponds to.

View the ODS Document Object Model (DOM) for PDF Output

``` SAS
options nodate nonumber obs=25;
ods html close;
ods pdf notoc file="myoutput.pdf" dom;

title "Comparison of City and Highway Miles Per Gallon";
title2 "Default Output";

proc print data=sashelp.cars;
   var make model type Origin MPG_City MPG_Highway;
run;

ods pdf close;
```

Program Description
Before you create your CSS file, you can view the ODS document object model for your output. You can use the DOM to decide what elements to use to customize areas of your output. For documentation about the ODS DOM, see Chapter 6, “Working with the ODS Document Object Model,” on page 45.

---

**Close the HTML destination and set the SAS system options.** The HTML destination is open by default. If you do not want to create HTML output, close the HTML destination. Closing the HTML destination saves system resources.

``` SAS
options nodate nonumber obs=25;
ods html close;
```

**Create PDF output and view the DOM.** The ODS PDF statement creates the PDF output. The DOM option specifies that the DOM is written to the SAS log.

``` SAS
ods pdf notoc file="myoutput.pdf" dom;
```

**Create PROC PRINT output and close the PDF destination.**

``` SAS
title "Comparison of City and Highway Miles Per Gallon";
title2 "Default Output";

proc print data=sashelp.cars;
   var make model type Origin MPG_City MPG_Highway;
run;

ods pdf close;
```

**PDF DOM Viewed in the SAS Log**
The following SAS log output contains the ODS DOM for the PDF output.
Log 9.1  SAS Log Output

```sas
options nodate nonumber obs=25;
ods html close;ods pdf nobookmarklist nobookmarkgen file="myoutput.pdf" dom;
NOTE: Writing ODS PDF output to DISK destination "your-path\myoutput.pdf", printer "PDF".
<DOCTYPE html>
<html>
<head>
<title>ODS PDF DOM</title>
<link rel="stylesheet" href="style.css">
</head>
<body dest="pdf" class="body">
title "Comparison of City and Highway Miles Per Gallon";
proc print data=sashelp.cars;
var make model type Origin MPG_City MPG_Highway;
run;
</body>
</html>
```

Example 1: Using Colors and Class Selectors to Highlight Output Areas

```
<section id="idx" class="oo" data-name="procprinttable" label="data set sashelp.cars"
proc="print" output="print" contents-label="data set sashelp.cars">
<table class="table">
<colgroup>
<col type="num" name="obs">
<col type="char" name="make">
<col type="char" name="model">
<col type="char" name="origin">
<col type="num" name="mpg_city">
<col type="num" name="mpg_highway">
</colgroup>
<thead>
<tr>
<th class="header" type="char" index="1" name="obs" data-name="obs" label="obs">Obs</th>
<th class="header" type="char" index="2" name="make" data-name="make" label="make">Make</th>
<th class="header" type="char" index="3" name="model" data-name="model" label="model">Model</th>
<th class="header" type="char" index="4" name="origin" data-name="origin" label="origin">Origin</th>
<th class="header" type="char" index="5" name="mpg_city" data-name="mpg_city" label="mpg_city">MPG_City</th>
<th class="header" type="char" index="6" name="mpg_highway" data-name="mpg_highway" label="mpg_highway">MPG_Highway</th>
</tr>
</thead>
```
1. The `<TITLE>` element contains text that tells you the DOM type. In this case, the DOM is PDF.

2. The `<BODY>` element that contains the `class="body"` attribute is used in the preceding CSS to change the style properties of the part of the output that the element corresponds to. In the preceding CSS, the class style selector that is used to reference this element is `.BODY`. All of the elements that contain the attribute `class="body"` have the style properties applied to them, unless they are overridden by a following rule set.

3. The `<H1>` element that contains the `class="systemtitle"` attribute is used in the preceding CSS to change the style properties of the part of the output that the element corresponds to. In the preceding CSS, the class style selector that is used to reference this element is `.SYSTEMTITLE`. All of the elements that contain the attribute `class="systemtitle"` have the style properties applied to them, unless they are overridden by a following rule set.

4. The `<H1>` element that contains the `class="systemtitle2"` attribute is used in the preceding CSS to change the style properties of the part of the output that the element corresponds to. In the preceding CSS, the class style selector that is used to reference this element is `.SYSTEMTITLE2`. All of the elements that contain the attribute `class="systemtitle2"` have the style properties applied to them, unless they are overridden by a following rule set.

5. The `<TABLE>` element is used in the preceding CSS to change the style properties of the part of the output that the element corresponds to. In the preceding CSS, the element style selector that is used to reference this element is `.TABLE`. Notice that because this is an element selector, there is no period before the element name. All of the `<TABLE>` elements have the style properties applied to them, unless they are overridden by a following rule set.

6. The `<TH>` element that contains the `class="header"` attribute is used in the preceding CSS to change the style properties of the part of the output that the element corresponds to. In the preceding CSS, the attribute style selector that is used to reference this element is `.HEADER`. All of the elements that contain the attribute `class="header"` have the style properties applied to them, unless they are overridden by a following rule set.
The `<TH>` element that contains the `data-name="obs"` attribute is used in the preceding CSS to change the style properties of the part of the output that the element corresponds to. In the preceding CSS, the attribute style selector that is used to reference this element is `[DATA-NAME="OBS"]`. All of the elements that contain the attribute `data-name="obs"` have the style properties applied to them, unless they are overridden by a following rule set.

The `<TD>` element that contains the `class="data"` attribute is used in the preceding CSS to change the style properties of the part of the output that the element corresponds to. In the preceding CSS, the class style selector that is used to reference this element is `.DATA`. All of the elements that contain the attribute `class="data"` have the style properties applied to them, unless they are overridden by a following rule set.

The `<TH>` element that contains the `unformatted-type="char"` attribute is used in the preceding CSS to change the style properties of the part of the output that the element corresponds to. In the preceding CSS, the attribute style selector that is used to reference this element is `[UNFORMATTED-TYPE="CHAR"]`. All of the elements that contain the attribute `unformatted-type="char"` have the style properties applied to them, unless they are overridden by a following rule set.

The `<TH>` element that contains the `name="mpg_city"` attribute is used in the preceding CSS to change the style properties of the part of the output that the element corresponds to. In the preceding CSS, the attribute style selector that is used to reference this element is `[NAME="MPG_CITY" ]`. All of the elements that contain the attribute `name="mpg_city"` have the style properties applied to them, unless they are overridden by a following rule set.

### Create the Cascading Style Sheet

The following is industry-standard CSS syntax that is based on the ODS HTML DOM. It is saved in an external file named ColorMapCss.css.

```css
.body {
  background-color: lightblue;
}

.systemtitle {
  background-color: darkblue;
  color: white;
}

.systemtitle2 {
  background-color: white;
}

table {
  background-color: white;
}

.header {
  background-color: gold;
}

[data-name="obs"] {
  background-color: purple;
  color: white;
}
```
Apply the CSS to PDF Output

```
options nodate nonumber obs=25;
ods html close;
ods pdf notoc file="myoutput.pdf"
   cssstyle="your-file-path\ColorMapCss.css";

title "Comparison of City and Highway Miles Per Gallon";
title2 "Colors Added by CSS";

proc print data=sashelp.cars;
   var make model type Origin MPG_City MPG_Highway;
run;
ods pdf close;
```

Program Description

After you have created your CSS, you can apply it to your PDF output by using the CSSSTYLE= option in the ODS PDF statement.

**Close the HTML destination and set the SAS system options.** The HTML destination is open by default. If you do not want to create HTML output, close the HTML destination. Closing the HTML destination saves system resources.

```
options nodate nonumber obs=25;
ods html close;
```

**Create PDF output and apply the CSS file.** The ODS PDF statement creates the PDF output. The CSSSTYLE= option specifies that the style sheet ColorMapCss.css is applied to your output.

```
ods pdf notoc file="myoutput.pdf"
   cssstyle="your-file-path\ColorMapCss.css";
```

**Create PROC PRINT output and close the PDF destination.**

```
title "Comparison of City and Highway Miles Per Gallon";
title2 "Colors Added by CSS";

proc print data=sashelp.cars;
```
Example 1: Using Colors and Class Selectors to Highlight Output Areas

```
var make model type Origin MPG_City MPG_Highway;
run;

ods pdf close;
```

Output 9.1  Default PDF Output

Comparison of City and Highway Miles Per Gallon
Default Output

<table>
<thead>
<tr>
<th>Obs</th>
<th>Make</th>
<th>Model</th>
<th>Type</th>
<th>Origin</th>
<th>MPG_City</th>
<th>MPG_Highway</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Acura</td>
<td>MDX</td>
<td>SUV</td>
<td>Asia</td>
<td>17</td>
<td>23</td>
</tr>
<tr>
<td>2</td>
<td>Acura</td>
<td>RSX Type 5 2dr</td>
<td>Sedan</td>
<td>Asia</td>
<td>24</td>
<td>31</td>
</tr>
<tr>
<td>3</td>
<td>Acura</td>
<td>TSX 4dr</td>
<td>Sedan</td>
<td>Asia</td>
<td>22</td>
<td>29</td>
</tr>
<tr>
<td>4</td>
<td>Acura</td>
<td>TL 4dr</td>
<td>Sedan</td>
<td>Asia</td>
<td>20</td>
<td>28</td>
</tr>
<tr>
<td>5</td>
<td>Acura</td>
<td>2.5 RL 4dr</td>
<td>Sedan</td>
<td>Asia</td>
<td>18</td>
<td>24</td>
</tr>
<tr>
<td>6</td>
<td>Acura</td>
<td>2.5 RL w/Navigation 4dr</td>
<td>Sedan</td>
<td>Asia</td>
<td>18</td>
<td>24</td>
</tr>
<tr>
<td>7</td>
<td>Acura</td>
<td>NSX coupe 2dr manual S</td>
<td>Sports</td>
<td>Asia</td>
<td>17</td>
<td>24</td>
</tr>
<tr>
<td>8</td>
<td>Audi</td>
<td>A3 1.8T 4dr</td>
<td>Sedan</td>
<td>Europe</td>
<td>22</td>
<td>31</td>
</tr>
<tr>
<td>9</td>
<td>Audi</td>
<td>A4 1.8T convertible 2dr</td>
<td>Sedan</td>
<td>Europe</td>
<td>23</td>
<td>30</td>
</tr>
<tr>
<td>10</td>
<td>Audi</td>
<td>A4 3.0 4dr</td>
<td>Sedan</td>
<td>Europe</td>
<td>20</td>
<td>28</td>
</tr>
<tr>
<td>11</td>
<td>Audi</td>
<td>A4 2.0 Quattro 4dr manual</td>
<td>Sedan</td>
<td>Europe</td>
<td>17</td>
<td>26</td>
</tr>
<tr>
<td>12</td>
<td>Audi</td>
<td>A4 2.0 Quattro 4dr auto</td>
<td>Sedan</td>
<td>Europe</td>
<td>16</td>
<td>25</td>
</tr>
<tr>
<td>13</td>
<td>Audi</td>
<td>A6 3.0 4dr</td>
<td>Sedan</td>
<td>Europe</td>
<td>20</td>
<td>27</td>
</tr>
<tr>
<td>14</td>
<td>Audi</td>
<td>A6 3.0 Quattro 4dr</td>
<td>Sedan</td>
<td>Europe</td>
<td>18</td>
<td>25</td>
</tr>
<tr>
<td>15</td>
<td>Audi</td>
<td>A4 3.0 convertible 2dr</td>
<td>Sedan</td>
<td>Europe</td>
<td>20</td>
<td>27</td>
</tr>
<tr>
<td>16</td>
<td>Audi</td>
<td>A4 3.0 Quattro convertible 2dr</td>
<td>Sedan</td>
<td>Europe</td>
<td>18</td>
<td>25</td>
</tr>
<tr>
<td>17</td>
<td>Audi</td>
<td>A6 2.7 Turbo Quattro 4dr</td>
<td>Sedan</td>
<td>Europe</td>
<td>16</td>
<td>26</td>
</tr>
<tr>
<td>18</td>
<td>Audi</td>
<td>A6 4.2 Quattro 4dr</td>
<td>Sedan</td>
<td>Europe</td>
<td>17</td>
<td>38</td>
</tr>
<tr>
<td>19</td>
<td>Audi</td>
<td>A8 L Quattro 4dr</td>
<td>Sedan</td>
<td>Europe</td>
<td>17</td>
<td>24</td>
</tr>
<tr>
<td>20</td>
<td>Audi</td>
<td>S4 Quattro 4dr</td>
<td>Sedan</td>
<td>Europe</td>
<td>14</td>
<td>20</td>
</tr>
<tr>
<td>21</td>
<td>Audi</td>
<td>RS 6 4dr</td>
<td>Sports</td>
<td>Europe</td>
<td>15</td>
<td>22</td>
</tr>
<tr>
<td>22</td>
<td>Audi</td>
<td>TT 1.8 convertible 2dr (coupe)</td>
<td>Sports</td>
<td>Europe</td>
<td>20</td>
<td>26</td>
</tr>
<tr>
<td>23</td>
<td>Audi</td>
<td>TT 1.8 Quattro 2dr (convertible)</td>
<td>Sports</td>
<td>Europe</td>
<td>20</td>
<td>28</td>
</tr>
<tr>
<td>24</td>
<td>Audi</td>
<td>TT 3.2 coupe 2dr (convertible)</td>
<td>Sports</td>
<td>Europe</td>
<td>21</td>
<td>29</td>
</tr>
<tr>
<td>25</td>
<td>Audi</td>
<td>A6 3.0 AvantQuattro</td>
<td>Wagon</td>
<td>Europe</td>
<td>18</td>
<td>25</td>
</tr>
</tbody>
</table>
Example 2: Applying Styles to Lists

Features:
- CSSSTYLE= option
- DOM option
- ODS HTML CLOSE statement
- ODS PDF statement

Other features:
- Industry-standard CSS style selectors
  - Class selectors
  - Pseudo-Class selectors
- Industry-standard CSS style properties
  - COLOR:
  - FONT-WEIGHT:
  - FONT-SIZE:
  - LIST-STYLE-TYPE:
- ODSLIST Procedure

Details

The following example shows you how to create a CSS that changes the style of list markers and list items. This example uses class selectors and pseudo-class selectors.

Class selectors are style selectors that select elements based on the value of the class= attribute in the element. Class selectors must have a period (.) preceding the class name. In this example, we use class selectors to change appearance of the content of the list items.

Pseudo-class selectors are style selectors that select elements based on the relationships between DOM elements. Pseudo-classes are represented by the pseudo-class name prefixed with a colon (:). In this example, we use pseudo-class selectors to change appearance of the marker of the list items.
Create Your Cascading Style Sheet

The following is industry-standard CSS syntax that is based on the ODS PDF DOM. It is saved in an external file named listStyle.css.

```css
.listitem, .listitem li::marker{
  color: blue;
  font-weight: bold;
  font-size: large;
  list-style-type: circle;
}

.listitem2, .listitem2 li::marker{
  color: red;
  font-size: medium;
  list-style-type: decimal;
}

.listitem3, .listitem3 li::marker{
  color: purple;
  list-style-type: circle;
}
```

View the ODS Document Model

```ods
ods html close;
options nodate nonumber;
title "Applying Styles to PRINTER Output";
ods pdf dom file="listStyle.pdf" ;
proc odslist;
  item;
    p ' Topics For This Week';
    list ;
    item ;
    p ' Fraud' ;
    list ;
      item 'Consumer Fraud' / value=1 ;
      item 'Business Fraud';
    end;
  end;
  item ' Customer Intelligence' ;
  item ' Social Media';
  item ' Data Mining';
  item ' High-Performance Computing';
  item ' Risk';
  item ' Data Management';
end;
end;
run;

title;
proc odslist;
  item;
    p ' Topics For Today';
```
Program Description

Before you create your CSS file, you can view the ODS document object model for your output. You can use the DOM to decide what elements to use to customize areas of your output. For documentation about the ODS DOM, see Chapter 6, “Working with the ODS Document Object Model,” on page 45.

Close the HTML destination, set the system options, and specify a title.

ods html close;
options nodate nonumber;
title "Applying Styles to PRINTER Output";

Write the DOM for your output to the SAS log. The DOM option in the ODS PDF statement specifies that the DOM is written to the SAS log.

ods pdf dom file="listStyle.pdf" ;

Create the PROC ODSSLIST output.

proc odslist;
  item;
  p ' Topics For This Week';
  list ;
  item ;

  p ' Fraud' ;
  list ;
  item 'Consumer Fraud' / value=1 ;
  item 'Business Fraud';
  end;
end;

item ' Customer Intelligence' ;
item ' Social Media';
item ' Data Mining';
item ' High-Performance Computing';
item ' Risk';
item ' Data Management';
end;
end;
run;

title;
proc odslist;
  item;
  p ' Topics For Today';
  list ;
  item ' Customer Intelligence';
PDF DOM Viewed in the SAS Log

The following partial SAS log output contains the ODS DOM for the PDF output.
1135 proc odslist;
1136   item;
1137   p ' Topics For This Week';
1138   list;
1139   item;
1140   p ' Fraud';
1141      list;
1142      item 'Consumer Fraud' / value=1;
1143      item 'Business Fraud';
1144   end;
1145 end;
1146
topicslist:
1147   item ' Customer Intelligence';
1148   item ' Social Media';
1149   item ' Data Mining';
1150   item ' High-Performance Computing';
1151   item ' Risk';
1152   item ' Data Management';
1153 end;
1154 end;
1155 run;
1156
1157 <h1 class="systemtitle">Applying Styles to PRINTER Output</h1>
1158
1159 <ul class="list">
1160   <li class="listitem" type="char">
1161     <li class="listitem" type="char">
1162       Topics For This Week
1163     </li>
1164   </li>
1165   <li class="listitem2" type="char">
1166     <li class="listitem2" type="char">
1167       Fraud
1168     </li>
1169   </li>
1170   <li class="listitem3" type="char">
1171     <li class="listitem3" type="char">
1172     </li>
1173   </li>
1174 </ul>

...more log output...
The `<LI>` element that contains the `class="listitem"` attribute is used in the preceding CSS to change the style properties of the list item contents for the highest-level list items. In the preceding CSS, the class selector that is used to reference this element is `.listitem`.

The `<LI::MARKER>` element that contains the `class="listitem1"` attribute is used in the preceding CSS to change the style properties of the list marker for the highest level list items. In the preceding CSS, the pseudo-style selector that is used to reference this element is `.listitem li::marker`.

The `<LI>` element that contains the `class="listitem2"` attribute is used in the preceding CSS to change the style properties of the list item contents for the second level list items. In the preceding CSS, the class selector that is used to reference this element is `.listitem2`.

The `<LI::MARKER>` element that contains the `class="listitem2"` attribute is used in the preceding CSS to change the style properties of the list marker for the second level list items. In the preceding CSS, the pseudo-style selector that is used to reference this element is `.listitem2 li::marker`.

The `<LI>` element that contains the `class="listitem3"` attribute is used in the preceding CSS to change the style properties of the list item contents for the third level list items. In the preceding CSS, the class selector that is used to reference this element is `.listitem3`.

The `<LI::MARKER>` element that contains the `class="listitem3"` attribute is used in the preceding CSS to change the style properties of the list marker for the third level list items. In the preceding CSS, the pseudo-style selector that is used to reference this element is `.listitem3 li::marker`.

Apply the CSS File to Your Output

```plaintext
ods html close;
options nodate nonumber;
title "Applying Styles to PRINTER Output";
ods pdf file="listStyle.pdf" cssstyle='your-path\listStyle.css';

proc odslist;
  item;
  p ' Topics For This Week';
  list ;
  item ;
    p ' Fraud';
    list ;
      item 'Consumer Fraud' / value=1 ;
      item 'Business Fraud';
    end;
  end;

  item ' Customer Intelligence' ;
  item ' Social Media';
  item ' Data Mining';
  item ' High-Performance Computing';
  item ' Risk';
  item ' Data Management';
end;
```
Program Description

After you have created your CSS, you can apply it to your PDF output by using the CSSSTYLE= option in the ODS PDF statement.

Close the HTML destination, set the system options, and specify a title.

ods html close;
options nodate nonumber;
title "Applying Styles to PRINTER Output";

Apply your CSS file to the output. The CSSSTYLE= option in the ODS PDF statement applies the CSS to your PDF output.

ods pdf file="listStyle.pdf" cssstyle='your-path\listStyle.css';

Create the PROC ODSLIST output.

proc odslist;
   item;
      p ' Topics For This Week';
      list ;
      item ;
      p ' Fraud';
      list ;
      item 'Consumer Fraud' / value=1 ;
      item 'Business Fraud';
      end;
   end;
item ' Customer Intelligence' ;
item ' Social Media';
item ' Data Mining';
item ' High-Performance Computing';
item ' Risk';
item ' Data Management';
end;
end;
run;
title;
proc odslist;
  item;
    p ' Topics For Today';
    list ;
    item ' Customer Intelligence';
    item ' Social Media';
    item ' Data Mining';
  end;
end;
run;

Output 9.3  Default PROC ODSLIST Output

Applying Styles to PRINTER Output

- Topics For This Week
  1. Fraud
     - Consumer Fraud
     - Business Fraud
  2. Customer Intelligence
  3. Social Media
  4. Data Mining
  5. High-Performance Computing
  6. Risk
  7. Data Management

- Topics For Today
  1. Customer Intelligence
  2. Social Media
  3. Data Mining

Output 9.4  PROC ODSLIST Output with CSS Applied

Applying Styles to PRINTER Output
Example 3: Applying Styles to Table Cells

Features:
- CSSSTYLE= option
- ODS HTML statement
- DOM option

Other features:
- Industry-standard CSS style selectors
  - Class selectors
  - pseudo-class selectors
- Industry-standard CSS style properties
  - BACKGROUND-COLOR:
  - BORDER:
  - COLOR:
  - FONT-FAMILY:
  - FONT-WEIGHT:
  - PADDING:

Details

The following example shows you how to create a CSS by looking at the DOM and selecting elements. This example uses class selectors and pseudo-class selectors. Class selectors are style selectors that select elements based on the value of the class= attribute in the element. Class selectors must have a period (.) preceding the class name.

Pseudo-class selectors are style selectors that select elements based on the relationships between DOM elements. Pseudo-classes are represented by the pseudo-class name prefixed with a colon (:). In this example, we use pseudo-class selectors to change the background color of the Total rows to red and the font color to white.

The pseudo-class element used in this example has the following form:

```
table tbody tr:nth-child(9) > *
```

The spaces between TABLE, TBODY, and TR:NTH-CHILD(9) are descendant combinators. They specify that SAS looks at the content within any <TR> element that is within a <TBODY> element that is within a <TABLE> element. The section TR:NTH-CHILD(9) > * specifies that the 9th child of all of the <TR> elements is selected. For information about combinators, see “Combinators” on page 53. For information about pseudo-class selectors, see “Pseudo-Class Selectors” on page 51.

View the ODS Document Model

```
ods html file="MyHtmlFile.html" dom;

proc sort data=sashelp.prdsale out=prdsale;
  by Country;
run;

title 'Actual Product Sales';
title2 '(Custom Style Sheet Applied)';

proc tabulate data=prdsale;
  class region division prodtype;
  var actual;
```
keylabel all='Total';
  table (region all)*(division all),  
    (prodtype all)*(actual*f=dollar10.)  / 
    misstext=[label='Missing'] 
    box=[label='Region by Division and Type'];
run;

ods _all_ close;
ods html;

Program Description

Before you create your CSS file, you can view the ODS document object model for your 
output. You can use the DOM to decide what elements to use to customize areas of your 
output. For documentation about the ODS DOM, see Chapter 6, “Working with the ODS 

Write the DOM for your output to the SAS log. The DOM option in the ODS HTML 
statement specifies that the DOM is written to the SAS log.
ods html file="MyHtmlFile.html" dom;

Create the PROC TABULATE output.

proc sort data=sashelp.prdsale out=prdsale;   
  by Country;  
run;

title 'Actual Product Sales';
title2 '(Custom Style Sheet Applied)';

proc tabulate data=prdsale;
  class region division prodtype;
  var actual;
  keylabel all='Total';
  table (region all)*(division all),
    (prodtype all)*(actual*f=dollar10.)  / 
    misstext=[label='Missing'] 
    box=[label='Region by Division and Type'];
run;

ods _all_ close;
ods html;

HTML DOM Viewed in the SAS Log

The following SAS log output contains the ODS DOM for the HTML output.
Log 9.2  SAS Log Output

```sas
ods html file="MyHtmlFile.html" dom;
NOTE: Writing HTML Body file: MyHtmlFile.html
<!DOCTYPE html>
<html>
<head>
  <title>ODS HTML4 DOM</title>
  <link rel="stylesheet" href="style.css">
</head>
<body dest="html4" class="body">
  ... more log output ...
</body>
</html>
```

```sas
section id="idx" class="oo" data-name="report" label="table 1" proc="tabulate"
output="table" contents-label="table 1" data-label="table 1">
  ... more log output ...
```

```sas
  Proc "tabulate"
  output="table" contents-label="table 1" data-label="table 1">
  ... more log output ...
```
<table>
<thead>
<tr>
<th>Region by Division and Type</th>
<th>Product type</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>FURNITURE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Actual Sales</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Actual Sales</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sum</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sum</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Example 3: Applying Styles to Table Cells
The `<TITLE>` element contains text that tells you the DOM type. In this case, the DOM is HTML4.

The `<BODY>` element that contains the class="body" attribute is used in the preceding CSS to change the style properties of the part of the output that the element corresponds to. In the preceding CSS, the class style selector that is used to reference this element is .BODY. All of the elements that contain the attribute class="body" have the style properties applied to them, unless they are overridden by a following rule set.
3 The `<H1>` element that contains the `class="systemtitle"` attribute is used in the preceding CSS to change the style properties of the part of the output that the element corresponds to. In the preceding CSS, the class style selector that is used to reference this element is `.SYSTEMTITLE`. All of the elements that contain the attribute `class="systemtitle"` have the style properties applied to them, unless they are overridden by a following rule set.

4 The `<TH>` element that contains the `class="header"` attribute is used in the preceding CSS to change the style properties of the part of the output that the element corresponds to. In the preceding CSS, the attribute style selector that is used to reference this element is `.HEADER`. All of the elements that contain the attribute `class="header"` have the style properties applied to them, unless they are overridden by a following rule set.

5 The `<TH>` element that contains the `class="rowheader"` attribute is used in the preceding CSS to change the style properties of the part of the output that the element corresponds to. In the preceding CSS, the class style selector that is used to reference this element is `.ROWHEADER`. All of the elements that contain the attribute `class="rowheader"` have the style properties applied to them, unless they are overridden by a following rule set.

6 The `<TD>` element that contains the `class="data"` attribute is used in the preceding CSS to change the style properties of the part of the output that the element corresponds to. In the preceding CSS, the class style selector that is used to reference this element is `.DATA`. All of the elements that contain the attribute `class="data"` have the style properties applied to them, unless they are overridden by a following rule set.

7 You can use the following rule set to change the color of specific cells in this table to red:

```
table tbody tr:nth-child(9) > *,
table tbody tr:nth-child(10) > *,
table tbody tr:nth-child(11) > *,
table tbody tr:nth-child(12) > *
    background-color: red;
    color: white;
}
```

This `<TABLE>` element corresponds to the `<TABLE>` element style selector in the preceding rule set.

**Create Your Cascading Style Sheet**

```
.body, .systemtitle, .header, .rowheader, .data {
    font-family: arial, sans-serif;
}

.body{
    color: black;
    background-color: gray;
}

.table, .header, .rowheader, .data {
    border: 1px solid #6699FF;
    padding: 5px 10px;
    font-family: arial, sans-serif;
    font-weight: bold;
}
```
Program Description

The following is industry-standard CSS syntax that is based on the ODS HTML DOM. It is saved in an external file named MyCss.css.
Select specific cells in the output to have a red background. The following style selectors set the background and font color of the cells in the 9th, 10th, 11th, and 12th rows in the section of the output controlled by the `<TBODY>` element.

```
table tbody tr:nth-child(9) > *,
table tbody tr:nth-child(10) > *,
table tbody tr:nth-child(11) > *,
table tbody tr:nth-child(12) > *
{
    background-color: red;
    color: white;
}
```

Apply the CSS File to Your Output

```
ods html file="MyHtmlFile.html" cssstyle='your-file-path/MyCss.css';
```

Program Description

After you have created your CSS, you can apply it to your HTML output by using the CSSSTYLE= option in the ODS HTML statement.

Apply your CSS file to the output. The CSSSTYLE= option in the ODS HTML statement specifies that the file MyCss.css is applied to your output.

```
ods html file="MyHtmlFile.html" cssstyle='your-file-path/MyCss.css';
```

Create the PROC TABULATE output.
proc sort data=sashelp.prdsale out=prdsale;
   by Country;
run;

title 'Actual Product Sales';
title2 '(Custom Style Sheet Applied)';

proc tabulate data=prdsale;
   class region division prodtype;
   var actual;
   keylabel all='Total';
   table (region all)*(division all),
               (prodtype all)*(actual*f=dollar10.) /
                misstext=[label='Missing']
                box=[label='Region by Division and Type'];
run;

ods _all_ close;
ods html;

Default Output

Output 9.5  Default HTML Output

<table>
<thead>
<tr>
<th>Region by Division and Type</th>
<th>Product type</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>FURNITURE</td>
<td>OFFICE</td>
</tr>
<tr>
<td></td>
<td>Actual Sales</td>
<td>Actual Sales</td>
</tr>
<tr>
<td></td>
<td>Sum</td>
<td>Sum</td>
</tr>
<tr>
<td>EAST</td>
<td>CONSUMER</td>
<td>$72,570</td>
</tr>
<tr>
<td></td>
<td>EDUCATION</td>
<td>$73,901</td>
</tr>
<tr>
<td></td>
<td>Total</td>
<td>$146,471</td>
</tr>
<tr>
<td>WEST</td>
<td>Division</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CONSUMER</td>
<td>$76,209</td>
</tr>
<tr>
<td></td>
<td>EDUCATION</td>
<td>$67,945</td>
</tr>
<tr>
<td></td>
<td>Total</td>
<td>$144,154</td>
</tr>
<tr>
<td>Total</td>
<td>Division</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CONSUMER</td>
<td>$148,779</td>
</tr>
<tr>
<td></td>
<td>EDUCATION</td>
<td>$141,846</td>
</tr>
<tr>
<td></td>
<td>Total</td>
<td>$290,625</td>
</tr>
</tbody>
</table>
### Example 4: Using Dynamic Property Values

**Features:**
- CSSSTYLE= option
- ODS PDF statement
- DOM option

**Other features:**
- Industry-standard CSS style selectors
  - Class selectors
  - Attribute selectors
- Industry-standard CSS style properties
  - BACKGROUND-COLOR:
  - BORDER-WIDTH:
  - COLOR:
  - FONT-SIZE:
  - FONT-WEIGHT:
  - PADDING:
  - TEXT-ALIGN:
- SAS dynamic variable values
  - FORMAT
  - RESOLVE
  - SYMGET

---

**Output Customized with CSS**

**Output 9.6 Customized HTML Output**

<table>
<thead>
<tr>
<th>Region</th>
<th>Division</th>
<th>Product type</th>
<th>Actual Sales</th>
<th>Actual Sales</th>
<th>Actual Sales</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>FURNITURE</td>
<td>Sum</td>
<td>Sum</td>
<td>Sum</td>
</tr>
<tr>
<td>EAST</td>
<td>CONSUMER</td>
<td>$72,570</td>
<td>$108,686</td>
<td>$181,256</td>
<td></td>
</tr>
<tr>
<td></td>
<td>EDUCATION</td>
<td>$73,901</td>
<td>$115,104</td>
<td>$189,005</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Total</td>
<td>$146,471</td>
<td>$223,790</td>
<td>$370,261</td>
<td></td>
</tr>
<tr>
<td>WEST</td>
<td>Division</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>CONSUMER</td>
<td>$76,209</td>
<td>$105,020</td>
<td>$181,229</td>
<td></td>
</tr>
<tr>
<td></td>
<td>EDUCATION</td>
<td>$67,945</td>
<td>$110,902</td>
<td>$178,847</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Total</td>
<td>$144,154</td>
<td>$215,922</td>
<td>$360,076</td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>Division</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>CONSUMER</td>
<td>$148,779</td>
<td>$213,706</td>
<td>$362,485</td>
<td></td>
</tr>
<tr>
<td></td>
<td>EDUCATION</td>
<td>$141,846</td>
<td>$226,006</td>
<td>$367,852</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Total</td>
<td>$290,625</td>
<td>$439,712</td>
<td>$730,337</td>
<td></td>
</tr>
</tbody>
</table>
With ODS and CSS styles, you can use SAS dynamic variables to specify expressions, macro variables, and SAS functions. The following example uses dynamic variables to customize PDF output. For complete documentation about the SAS macro language, see *SAS Macro Language: Reference*. For complete documentation about PROC FORMAT, see *Base SAS Procedures Guide*.

**View the ODS Document Object Model (DOM) for PDF Output**

```sas
options nodate nonumber obs=25;
ods html close;
%let color = black;
%macro mpgC;
   %if &_val_ >=25 %then 15pt;
   %else 5pt;
%mend;
%macro mpgH;
   %if &_val_ >=40 %then 15pt;
   %else 5pt;
%mend;
proc format;
   value mpgC 25-108 ='#b1f100';
   value mpgH 41-118 ='yellow';
run;
proc sort data=sashelp.cars out=cars;
   by descending make;
run;
ods pdf notoc file="myoutput.pdf" dom;

title "City MPG=25-108 Shaded in Green with Large Font";
title2 "Highway MPG=41-118 Shaded in Yellow with Large Font";
proc print data=cars label;
   var make model type Origin MPG_City MPG_Highway;
run;
ods pdf close;
```

**Program Description**

Before you create your CSS file, you can view the ODS document object model for your output. You can use the DOM to decide what elements to use to customize areas of your output. For documentation about the ODS DOM, see Chapter 6, “Working with the ODS Document Object Model,” on page 45.

**Close the HTML destination and set the SAS system options.** The HTML destination is open by default. If you do not want to create HTML output, close the HTML destination. Closing the HTML destination saves system resources.

```sas
options nodate nonumber obs=25;
ods html close;
```
Create the macro variable Color. The %LET macro statement creates the macro variable Color and assigns it the color black.

```
%let color = black;
```

Create the macros mpgC and mpgH. The %MACRO macro statement creates the macros mpgC and mpgH. The built-in variable _VAL_ is the data value of the current cell. The macro mpgC changes the font to 15pt if the value of _VAL_ is greater than or equal to 25. The macro mpgH changes the font to 15pt if the value of _VAL_ is greater than or equal to 40.

```
%macro mpgC;
  %if &_val_ >=25 %then 15pt;
  %else 5pt;
%mend;

%macro mpgH;
  %if &_val_ >=40 %then 15pt;
  %else 5pt;
%mend;
```

Create the formats mpgC. and mpgH.. PROC FORMAT creates the formats mpgC. and mpgH.. The format mpgC. changes the color to #b1f100 if the value of the cell is between 25 and 108. The format mpgH. changes the color to yellow if the value of the cell is between 45 and 118.

```
proc format;
  value mpgC 25-108 ='#b1f100';
  value mpgH 41-118 = 'yellow';
run;
```

Sort the data.

```
proc sort data=sashelp.cars out=cars;
  by descending make;
run;
```

Create PDF output and view the DOM. The ODS PDF statement creates the PDF output. The DOM option specifies that the DOM is written to the SAS log.

```
ods pdf notoc file="myoutput.pdf" dom;
```

Specify the titles, create PROC PRINT output, and close the PDF destination.

```
title "City MPG=25-108 Shaded in Green with Large Font";
title2 "Highway MPG=41-118 Shaded in Yellow with Large Font";
proc print data=cars label;
  var make model type Origin MPG_City MPG_Highway;
run;
ods pdf close;
```

PDF DOM Viewed in the SAS Log

The following SAS log output contains the ODS DOM for the PDF output.
Log 9.3  SAS Log Output

...more log output...

```html
<html>
<head>
  <title>ODS PDF DOM</title>
  <link rel="stylesheet" href="style.css">
</head>
<body dest="pdf" class="body">
  71   title "City MPG=25-108 Shaded in Green with Large Font";
  72   title2 "Highway MPG=41-118 Shaded in Yellow with Large Font";
  73
  74   proc print data=cars label;
  75      var make model type Origin MPG_City MPG_Highway;
  76
  77   run;

  <section id="idx" class="oo" data-name="procprinttable" label="data set work.cars">
    proc="print" output="print" contents-label="data set work.cars">
  79      <h1 class="systemtitle">City MPG=25-108 Shaded in Green with Large Font</h1>
  80      <h1 class="systemtitle2">Highway MPG=41-118 Shaded in Yellow with Large Font</h1>
  <table class="table">
    <colgroup>
      <col type="num" name="obs">
      <col type="char" name="make">
      <col type="char" name="model">
      <col type="char" name="type">
      <col type="char" name="origin">
      <col type="num" name="mpg_city">
      <col type="num" name="mpg_highway">
    </colgroup>
    <thead>
      <tr>
        <th class="header" type="char" unformatted-type="char" index="1" name="obs" data-name="obs" label="obs">Obs</th>
        <th class="header" type="char" unformatted-type="char" index="2" name="make" data-name="make" label="make">Make</th>
        <th class="header" type="char" unformatted-type="char" index="3" name="model" data-name="model" label="model">Model</th>
      </tr>
    </thead>
    <tbody>
      ...more log output...
    </tbody>
  </table>
</section>
</body>
</html>
```
The `<TITLE>` element contains text that tells you the DOM type. In this case, the DOM is PDF.

The `<BODY>` element that contains the `class="body"` attribute is used in the preceding CSS to change the style properties of the part of the output that the element corresponds to. In the preceding CSS, the class style selector that is used to reference this element is .BODY. All of the elements that contain the attribute `class="body"` have the style properties applied to them, unless they are overridden by a following rule set.

The `<H1>` element that contains the `class="systemtitle"` attribute is used in the preceding CSS to change the style properties of the part of the output that the element corresponds to. In the preceding CSS, the class style selector that is used to
reference this element is .SYSTEMTITLE. All of the elements that contain the attribute class="systemtitle" have the style properties applied to them, unless they are overridden by a following rule set.

4 The <H1> element that contains the class="systemtitle2" attribute is used in the preceding CSS to change the style properties of the part of the output that the element corresponds to. In the preceding CSS, the class style selector that is used to reference this element is .SYSTEMTITLE2. All of the elements that contain the attribute class="systemtitle2" have the style properties applied to them, unless they are overridden by a following rule set.

5 The <TABLE> element is used in the preceding CSS to change the style properties of the part of the output that the element corresponds to. In the preceding CSS, the element style selector that is used to reference this element is .TABLE. All of the <TABLE> elements have the style properties applied to them, unless they are overridden by a following rule set.

6 The <TH> element that contains the class="header" attribute is used in the preceding CSS to change the style properties of the part of the output that the element corresponds to. In the preceding CSS, the attribute style selector that is used to reference this element is .HEADER. All of the elements that contain the attribute class="header" have the style properties applied to them, unless they are overridden by a following rule set.

7 The <TH> element that contains the data-name="obs" attribute is used in the preceding CSS to change the style properties of the part of the output that the element corresponds to. In the preceding CSS, the attribute style selector that is used to reference this element is [DATA-NAME="OBS"]. All of the elements that contain the attribute data-name="obs" have the style properties applied to them, unless they are overridden by a following rule set.

8 The <TH> element that contains the class="rowheader" attribute is used in the preceding CSS to change the style properties of the part of the output that the element corresponds to. In the preceding CSS, the attribute style selector that is used to reference this element is .ROWHEADER. All of the elements that contain the attribute class="rowheader" have the style properties applied to them, unless they are overridden by a following rule set.

9 The <TD> element that contains the class="data" attribute is used in the preceding CSS to change the style properties of the part of the output that the element corresponds to. In the preceding CSS, the class style selector that is used to reference this element is .DATA. All of the elements that contain the attribute class="data" have the style properties applied to them, unless they are overridden by a following rule set.

10 The <TD> element that contains the name="mpg_city" attribute is used in the preceding CSS to change the style properties of the part of the output that the element corresponds to. In the preceding CSS, the attribute style selector that is used to reference this element is [NAME*="CITY" ]. The *= combinator specifies that the value of the attribute name= should end with the text string CITY. Elements that match this criteria have the style properties applied to them, unless they are overridden by a following rule set.

11 The <TD> element that contains the name="mpg_highway" attribute is used in the preceding CSS to change the style properties of the part of the output that the element corresponds to. In the preceding CSS, the attribute style selector that is used to reference this element is [NAME*="HIGHWAY" ]. The *= combinator specifies that the value of the attribute name= should end with the text string HIGHWAY.
Elements that match this criteria have the style properties applied to them, unless they are overridden by a following rule set.

Create the Cascading Style Sheet

```css
.body {
  background-color: lightblue;
}

.systemtitle, .systemtitle2 {
  background-color: darkblue;
  color: white;
  font-size: 12pt;
  font-weight: bold;
}

.table {
  background-color: white;
}

.header {
  font-size: 10pt;
  font-weight: bold;
  color: blue;
  background-color: gold;
}

.rowheader {
  background-color: purple;
  color: white;
}

.data {
  background-color: beige;
}

.header, .rowheader, .data {
  border-width: 1px;
  padding: 10px;
  text-align: center;
}

[name*="city"] {
  color: symget('color');
  font-size: resolve('mpgC');
  background-color: format('mpgC.\');
}

[name*="highway"] {
  color: symget('color');
  font-size: resolve('mpgH');
  background-color: format('mpgH.\');
}
```
Program Description

The following CSS rule sets are saved in an external file named MpgStyle.css.

```
.body {
  background-color: lightblue;
}

.systemtitle, .systemtitle2 {
  background-color: darkblue;
  color: white;
  font-size: 12pt;
  font-weight: bold;
}

.table {
  background-color: white;
}

.header {
  font-size: 10pt;
  font-weight: bold;
  color: blue;
  background-color: gold;
}

.rowheader {
  background-color: purple;
  color: white;
}

.data {
  background-color: beige;
}

.header, .rowheader, .data {
  border-width: 1px;
  padding: 10px;
  text-align: center;
}
```

**Resolve the variables.** The SYMGET property value resolves the macro variable Color. The RESOLVE property value resolves the macro %mpgC. The FORMAT property value resolves the format mpgC.

```
[name*="city"] {
  color: symget('color');
  font-size: resolve('%mpgC');
  background-color: format('mpgC.');
}
```

**Resolve the variables.** The SYMGET property value resolves the macro variable Color. The RESOLVE property value resolves the macro %mpgH. The FORMAT property value resolves the format mpgH.

```
[name*="highway"] {
```
Apply the CSS to PDF Output

```sas
options nodate nonumber;
odds html close;
%let color = black;
%macro mpgC;
  %if &_val_ >=25 %then 15pt;
  %else 5pt;
%mend;
%macro mpgH;
  %if &_val_ >=40 %then 15pt;
  %else 5pt;
%mend;

proc format;
  value mpgC 25-108 ='#b1f100';
  value mpgH 41-118 ='yellow';
run;
proc sort data=sashelp.cars out=cars;
  by descending make;
run;
odds pdf nobookmarklist nobookmarkgen file="myoutput.pdf"
cssstyle="MpgStyle.css";
title "City MPG=25-108 Shaded in Green with Large Font";
title2 "Highway MPG=41-118 Shaded in Yellow with Large Font";
proc print data=cars label;
  var make model type Origin MPG_City MPG_Highway;
run;
odds pdf close;
```

Program Description

After you have created your CSS, you can apply it to your PDF output by using the CSSSTYLE= option in the ODS PDF statement.

Close the HTML destination and set the SAS system options. The HTML destination is open by default. If you do not want to create HTML output, close the HTML destination. Closing the HTML destination saves system resources.

```sas
options nodate nonumber;
odds html close;
```

Create the macro variable Color. The %LET macro statement creates the macro variable Color and assigns it the color black.

```sas
%let color = black;
```
Create the macros mpgC and mpgH. The `%MACRO` macro statement creates the macros mpgC and mpgH. The built-in variable _VAL_ contains the data value of the current cell. The macro mpgC changes the font to 15pt if the value of _VAL_ is greater than or equal to 25. The macro mpgH changes the font to 15pt if the value of _VAL_ is greater than or equal to 40.

```latex
%macro mpgC;
  %if &_val_ >=25 %then 15pt;
  %else 5pt;
%mend;

%macro mpgH;
  %if &_val_ >=40 %then 15pt;
  %else 5pt;
%mend;
```

Create the formats mpgC. and mpgH.. PROC FORMAT creates the formats mpgC. and mpgH.. The format mpgC. changes the color to #b1f100 if the value of the cell is between 25 and 108. The format mpgH. changes the color to yellow if the value of the cell is between 45 and 118.

```latex
proc format;
  value mpgC 25-108 ='#b1f100';
  value mpgH 41-118 ='yellow';
run;
```

Sort the data.

```latex
proc sort data=sashelp.cars out=cars;
  by descending make;
run;
```

Create PDF output and specify the CSS file to be applied. The ODS PDF statement creates the PDF output. The CSSSTYLE= option specifies that the CSS file MpgStyle.css is applied to the output.

```latex
ods pdf nobookmarklist nobookmarkgen file="myoutput.pdf"
  cssstyle="MpgStyle.css";
```

Specify the titles, create PROC PRINT output, and close the PDF destination.

```latex
title "City MPG=25-108 Shaded in Green with Large Font";
title2 "Highway MPG=41-118 Shaded in Yellow with Large Font";

proc print data=cars label;
  var make model type Origin MPG_City MPG_Highway;
run;
ods pdf close;
```
### Example 4: Using Dynamic Property Values

**Output 9.7  Default PDF Output**

#### Comparison of City and Highway Miles Per Gallon

<table>
<thead>
<tr>
<th>Obs</th>
<th>Make</th>
<th>Model</th>
<th>Type</th>
<th>Origin</th>
<th>Mpg_City</th>
<th>Mpg_Highway</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Acura</td>
<td>MDX</td>
<td>SUV</td>
<td>Asia</td>
<td>17</td>
<td>23</td>
</tr>
<tr>
<td>2</td>
<td>Acura</td>
<td>RSX Type S 2dr</td>
<td>Sedan</td>
<td>Asia</td>
<td>24</td>
<td>31</td>
</tr>
<tr>
<td>3</td>
<td>Acura</td>
<td>TSX 4dr</td>
<td>Sedan</td>
<td>Asia</td>
<td>22</td>
<td>29</td>
</tr>
<tr>
<td>4</td>
<td>Acura</td>
<td>TL 4dr</td>
<td>Sedan</td>
<td>Asia</td>
<td>20</td>
<td>28</td>
</tr>
<tr>
<td>5</td>
<td>Acura</td>
<td>3.5 RL 4dr</td>
<td>Sedan</td>
<td>Asia</td>
<td>18</td>
<td>24</td>
</tr>
<tr>
<td>6</td>
<td>Acura</td>
<td>3.5 RL w/Navi 4dr</td>
<td>Sedan</td>
<td>Asia</td>
<td>18</td>
<td>24</td>
</tr>
<tr>
<td>7</td>
<td>Acura</td>
<td>NSX coupe 2dr manual S</td>
<td>Sports</td>
<td>Asia</td>
<td>17</td>
<td>24</td>
</tr>
<tr>
<td>8</td>
<td>Audi</td>
<td>A4 1.8T 4dr</td>
<td>Sedan</td>
<td>Europe</td>
<td>22</td>
<td>31</td>
</tr>
<tr>
<td>9</td>
<td>Audi</td>
<td>A4 1.8T convertible 2dr</td>
<td>Sedan</td>
<td>Europe</td>
<td>23</td>
<td>30</td>
</tr>
<tr>
<td>10</td>
<td>Audi</td>
<td>A4 2.0 4dr</td>
<td>Sedan</td>
<td>Europe</td>
<td>20</td>
<td>28</td>
</tr>
<tr>
<td>11</td>
<td>Audi</td>
<td>A4 2.0 Quattro 4dr</td>
<td>Sedan</td>
<td>Europe</td>
<td>17</td>
<td>26</td>
</tr>
<tr>
<td>12</td>
<td>Audi</td>
<td>A4 2.0 Quattro 4dr</td>
<td>Sedan</td>
<td>Europe</td>
<td>16</td>
<td>25</td>
</tr>
<tr>
<td>13</td>
<td>Audi</td>
<td>A6 3.0 4dr</td>
<td>Sedan</td>
<td>Europe</td>
<td>20</td>
<td>27</td>
</tr>
<tr>
<td>14</td>
<td>Audi</td>
<td>A6 3.0 Quattro 4dr</td>
<td>Sedan</td>
<td>Europe</td>
<td>18</td>
<td>25</td>
</tr>
<tr>
<td>15</td>
<td>Audi</td>
<td>A4 2.0 Quattro 2dr</td>
<td>Sedan</td>
<td>Europe</td>
<td>20</td>
<td>27</td>
</tr>
<tr>
<td>16</td>
<td>Audi</td>
<td>A4 2.0 Quattro convertible 2dr</td>
<td>Sedan</td>
<td>Europe</td>
<td>16</td>
<td>25</td>
</tr>
<tr>
<td>17</td>
<td>Audi</td>
<td>A6 2.0 Quattro 4dr</td>
<td>Sedan</td>
<td>Europe</td>
<td>18</td>
<td>25</td>
</tr>
<tr>
<td>18</td>
<td>Audi</td>
<td>A6 2.0 Quattro 4dr</td>
<td>Sedan</td>
<td>Europe</td>
<td>17</td>
<td>24</td>
</tr>
<tr>
<td>19</td>
<td>Audi</td>
<td>A8 L Quattro 4dr</td>
<td>Sedan</td>
<td>Europe</td>
<td>17</td>
<td>24</td>
</tr>
<tr>
<td>20</td>
<td>Audi</td>
<td>S4 Quattro 4dr</td>
<td>Sedan</td>
<td>Europe</td>
<td>14</td>
<td>20</td>
</tr>
<tr>
<td>21</td>
<td>Audi</td>
<td>RS 6 4dr</td>
<td>Sports</td>
<td>Europe</td>
<td>15</td>
<td>22</td>
</tr>
<tr>
<td>22</td>
<td>Audi</td>
<td>TT 1.8 convertible 2dr (coupe)</td>
<td>Sports</td>
<td>Europe</td>
<td>20</td>
<td>28</td>
</tr>
<tr>
<td>23</td>
<td>Audi</td>
<td>TT 1.8 Quattro 2dr (convertible)</td>
<td>Sports</td>
<td>Europe</td>
<td>20</td>
<td>28</td>
</tr>
<tr>
<td>24</td>
<td>Audi</td>
<td>TT 3.2 coupe 2dr (convertible)</td>
<td>Sports</td>
<td>Europe</td>
<td>21</td>
<td>29</td>
</tr>
<tr>
<td>25</td>
<td>Audi</td>
<td>A6 3.0 Avant Quattro</td>
<td>Wagon</td>
<td>Europe</td>
<td>18</td>
<td>25</td>
</tr>
</tbody>
</table>
# Table of Gas Mileage

<table>
<thead>
<tr>
<th>Make</th>
<th>Model</th>
<th>Type</th>
<th>Origin</th>
<th>City (MPG)</th>
<th>Highway (MPG)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Volvo</td>
<td>XC90 T8</td>
<td>SUV</td>
<td>Europe</td>
<td>25</td>
<td>10</td>
</tr>
<tr>
<td>Volvo</td>
<td>S40 4dr</td>
<td>Sedan</td>
<td>Europe</td>
<td>22</td>
<td>20</td>
</tr>
<tr>
<td>Volvo</td>
<td>S60 2.5 4dr</td>
<td>Sedan</td>
<td>Europe</td>
<td>26</td>
<td>25</td>
</tr>
<tr>
<td>Volvo</td>
<td>S60 T5 4dr</td>
<td>Sedan</td>
<td>Europe</td>
<td>28</td>
<td>25</td>
</tr>
<tr>
<td>Volvo</td>
<td>S80 2.9 4dr</td>
<td>Sedan</td>
<td>Europe</td>
<td>30</td>
<td>28</td>
</tr>
<tr>
<td>Volvo</td>
<td>S80 4.0 4dr</td>
<td>Sedan</td>
<td>Europe</td>
<td>33</td>
<td>30</td>
</tr>
<tr>
<td>Volvo</td>
<td>C70 LFT convertible 2dr</td>
<td>Sedan</td>
<td>Europe</td>
<td>34</td>
<td>30</td>
</tr>
<tr>
<td>Volvo</td>
<td>C70 HFT convertible 2dr</td>
<td>Sedan</td>
<td>Europe</td>
<td>35</td>
<td>30</td>
</tr>
<tr>
<td>Volvo</td>
<td>S80 V6 4dr</td>
<td>Sedan</td>
<td>Europe</td>
<td>36</td>
<td>30</td>
</tr>
<tr>
<td>Volvo</td>
<td>V40</td>
<td>Wagon</td>
<td>Europe</td>
<td>37</td>
<td>30</td>
</tr>
<tr>
<td>Volvo</td>
<td>XC70</td>
<td>Wagon</td>
<td>Europe</td>
<td>38</td>
<td>40</td>
</tr>
<tr>
<td>Volkswagen</td>
<td>Touareg V6</td>
<td>SUV</td>
<td>Europe</td>
<td>39</td>
<td>30</td>
</tr>
<tr>
<td>Volkswagen</td>
<td>Golf GLI 4dr</td>
<td>Sedan</td>
<td>Europe</td>
<td>40</td>
<td>30</td>
</tr>
<tr>
<td>Volkswagen</td>
<td>GTI 1.8T 2dr hatch</td>
<td>Sedan</td>
<td>Europe</td>
<td>41</td>
<td>30</td>
</tr>
<tr>
<td>Volkswagen</td>
<td>Jetta GLI TDI 4dr</td>
<td>Sedan</td>
<td>Europe</td>
<td>42</td>
<td>46</td>
</tr>
<tr>
<td>Volkswagen</td>
<td>Jetta GLI VRS 4dr</td>
<td>Sedan</td>
<td>Europe</td>
<td>43</td>
<td>40</td>
</tr>
<tr>
<td>Volkswagen</td>
<td>New Beetle GLS 1.8L 2dr</td>
<td>Sedan</td>
<td>Europe</td>
<td>44</td>
<td>30</td>
</tr>
<tr>
<td>Volkswagen</td>
<td>New Beetle GLS convertible 2dr</td>
<td>Sedan</td>
<td>Europe</td>
<td>45</td>
<td>30</td>
</tr>
<tr>
<td>Volkswagen</td>
<td>Passat GLS 4dr</td>
<td>Sedan</td>
<td>Europe</td>
<td>46</td>
<td>30</td>
</tr>
<tr>
<td>Volkswagen</td>
<td>Passat GLS V6 4motion 4dr</td>
<td>Sedan</td>
<td>Europe</td>
<td>47</td>
<td>30</td>
</tr>
<tr>
<td>Volkswagen</td>
<td>Passat V6 4motion 4dr</td>
<td>Sedan</td>
<td>Europe</td>
<td>48</td>
<td>30</td>
</tr>
<tr>
<td>Volkswagen</td>
<td>Phaeton V6 4dr</td>
<td>Sedan</td>
<td>Europe</td>
<td>49</td>
<td>30</td>
</tr>
<tr>
<td>Volkswagen</td>
<td>Phaeton V6 4dr</td>
<td>Sedan</td>
<td>Europe</td>
<td>50</td>
<td>30</td>
</tr>
<tr>
<td>Volkswagen</td>
<td>Jetta GL</td>
<td>Wagon</td>
<td>Europe</td>
<td>51</td>
<td>30</td>
</tr>
<tr>
<td>Volkswagen</td>
<td>Passat GLI 1.8T</td>
<td>Wagon</td>
<td>Europe</td>
<td>52</td>
<td>30</td>
</tr>
<tr>
<td>Volkswagen</td>
<td>Passat V8</td>
<td>Wagon</td>
<td>Europe</td>
<td>53</td>
<td>30</td>
</tr>
<tr>
<td>Toyota</td>
<td>Prius 4dr (gas/electric)</td>
<td>Hybrid</td>
<td>Asia</td>
<td>54</td>
<td>51</td>
</tr>
<tr>
<td>Toyota</td>
<td>Sequoia 5dr</td>
<td>SUV</td>
<td>Asia</td>
<td>55</td>
<td>30</td>
</tr>
<tr>
<td>Toyota</td>
<td>Highlander V6</td>
<td>SUV</td>
<td>Asia</td>
<td>56</td>
<td>34</td>
</tr>
<tr>
<td>Toyota</td>
<td>Land Cruiser</td>
<td>SUV</td>
<td>Asia</td>
<td>57</td>
<td>30</td>
</tr>
<tr>
<td>Toyota</td>
<td>RAV4</td>
<td>SUV</td>
<td>Asia</td>
<td>58</td>
<td>30</td>
</tr>
<tr>
<td>Toyota</td>
<td>Corolla CE 4dr</td>
<td>Sedan</td>
<td>Asia</td>
<td>59</td>
<td>40</td>
</tr>
<tr>
<td>Toyota</td>
<td>Corolla S 4dr</td>
<td>Sedan</td>
<td>Asia</td>
<td>60</td>
<td>40</td>
</tr>
<tr>
<td>Toyota</td>
<td>Corolla LE 4dr</td>
<td>Sedan</td>
<td>Asia</td>
<td>61</td>
<td>40</td>
</tr>
<tr>
<td>Toyota</td>
<td>Echo 2dr manual</td>
<td>Sedan</td>
<td>Asia</td>
<td>62</td>
<td>43</td>
</tr>
<tr>
<td>Toyota</td>
<td>Echo 2dr automatic</td>
<td>Sedan</td>
<td>Asia</td>
<td>63</td>
<td>35</td>
</tr>
</tbody>
</table>

**Output 9.8**  PDF Output with CSS Applied
Part 3

Report Writing Interface

Chapter 10

Introduction to the Report Writing Interface .......................... 111

Chapter 11

Reference and Examples .................................................. 121

Chapter 12

Concepts ........................................................................... 249
Overview

The traditional method for creating a custom report without creating a data set is called DATA_NULL report writing. With the ODS Report Writing Interface (RWI), you can create highly customized reports in an object-oriented language that is fully integrated with ODS. The ODS RWI does the following:

- fully embraces ODS features such as proportional fonts, trafficlighting, colors, images, and Unicode characters, while at the same time providing pixel-perfect placement capabilities
- takes advantage of the programming features that the DATA step offers, such as conditional logic, formatting capabilities, BY-group processing, and arrays
- uses an object-oriented language that provides you with flexibility and control so that even the most rigid reporting requirements can be met easily
Terminology

cell
is a collection of data, text, or images that can span multiple rows or columns.

cell padding
is the thickness on each of the four sides of the cell.

cell spacing
is the thickness of the spacing between cells.

dimension
is a positive integer followed by an optional unit of measure. A number without a unit of measure is interpreted as pixels.

TIP It is not recommended that you size things in pixels because of adverse dependencies on resolution that can differ between destinations.

class
is a template for an object. A class includes data that describes the object's characteristics (such as attributes or instance variables), as well as the operations (methods) that the object can perform.

column
is the vertical dimension of a 2-D arrangement (a collection of cells in the vertical dimension).

footnote
is descriptive text that appears at the bottom of your output. Title and footnote processing is always done before any output is produced on the physical page. This is often referred to as page initialization time. Footnotes can be specified inside regions, but only the footnotes that have changed will be displayed.

instance
is the run-time initialization of the class object attributes and methods.

layout container
is an area that contains a collection of regions. Layouts can contain only regions. Layouts can have a fixed size (such as width=3in and height=4in) or can be dynamically sized to accommodate the regions.

method
in object-oriented methodology, is an operation that is defined for a class and which can be executed by an object that is created from that class.

object
is any entity that can be manipulated by the commands of a programming language. Examples are values, variables, functions, and data structures.

region container
is an area that contains output (such as text, tables, or images), or nested layout containers. Regions can also have a fixed size, or might be dynamically sized to accommodate the collection of output. Regions cannot contain titles.

row
is the horizontal dimension of a 2-D arrangement (a collection of cells in the horizontal dimension).
Types of Methods

Overview of Report Writing Methods

There are various methods that make up the ODS Report Writing Interface. They can be organized into the following types:

• general purpose methods
• layout methods
• media methods
• page methods
• table methods
• text methods

CAUTION:
Not all destinations support the same set of features, and therefore some methods might have restrictions in some destinations.

General Purpose Methods

General purpose methods can be used to create URLs, insert images and lines into your output, or delete class instances.

The following are general purpose methods:

• “DELETE Method” on page 142
• “HREF Method” on page 157
• “LINE Method” on page 171

Layout Methods

Concepts

Layout methods control the placement of output on a page. There are two groups of layout methods, absolute and gridded. Layout containers can be nested, and gridded and absolute layouts can be used together. These terms frequently appear in layout documentation:
dimension unit
  is a nonnegative number followed by an optional unit of measure. A number without
  a unit of measure is interpreted as pixels. It is not recommended that you size things
  in pixels because of adverse dependencies on resolution that can differ between
  destinations.

footnote
  is descriptive text that appears at the bottom of your output. Title and footnote
  processing is always done before any output is produced on the physical page. This
  is often referred to as page initialization time. Footnotes can be specified inside
  regions, but only the footnotes that have changed will be displayed.

layout container
  is an area that contains a collection of regions. Layouts can contain only regions.
  Layouts can have a fixed size (like width=3in and height=4in) or can be dynamically
  sized to accommodate the regions.

region container
  is an area that contains output (like text, tables, images), or nested layout containers.
  Regions can also have a fixed size, or can be dynamically sized to accommodate the
  collection of output. Regions cannot contain titles.

title
  is descriptive text that appears at the top of your output. Title and footnote
  processing is always done before any output is produced on the physical page. This
  is often referred to as page initialization time.
Gridded Layout Methods

Gridded layout methods enable you to arrange output in a two-dimensional gridded structure, such as a spreadsheet or piece of graph paper. Gridded layout is a simple, powerful tool for arranging output and managing output on one or more pages. Gridded layouts enable you to do the following:

- enforce automatic alignment of respective grid cells
- continue onto the next page if necessary
- dynamically compute the size of a grid cell

The following are gridded layout methods:

- “LAYOUT_GRIDDED Method” on page 165
- “LAYOUT_END Method” on page 171
- “REGION Method, Gridded” on page 180

Absolute Layout Methods

Absolute layout methods enable you to specify the exact location of a layout and region container on a page. This method is very precise. Each container must be explicitly
placed to ensure no overlap. Absolute layouts are restricted to a single page. If the output is too large to fit in the fixed-size container, the output is discarded, and you receive a blank region and a warning in your log. Absolute layouts are suitable for static types of output, as in the following situations:

- placing output in a specific location on a pre-printed form
- creating cover pages
- precisely placing output in a nested region container

Absolute layouts are valid in printer family destinations only.

The following example uses absolute layout methods:

- “Example 12: Creating a Cover Page with the LAYOUT_ABSOLUTE Method” on page 243

The following are absolute layout methods:

- “LAYOUT_ABSOLUTE Method” on page 161
- “LAYOUT_END Method” on page 171
- “REGION Method, Absolute” on page 176

**Media Methods**

Media methods create audio and video files for HTML5 or EPUB output, or insert images into your output.

- “VIDEO Method” on page 191
- “AUDIO Method” on page 132
- “IMAGE Method” on page 160

**Page Methods**

Page methods control characteristics of the entire page. You can start a new page or modify an existing title or footnote. The TITLE and FOOTNOTE methods enable you to incorporate a particular data value into your title or footnote. These methods work similarly to the global TITLE and FOOTNOTE statements available outside of the DATA step.

The following example uses the PAGE method:

- “Example 10: Using the Page Method” on page 228

The following are page methods:

- “PAGE Method” on page 176
- “TITLE Method” on page 188
- “FOOTNOTE Method” on page 152

**Table Methods**

Table methods create tabular output that consists of row headings, column headings, and data values in a structured format. The ODS Report Writing Interface provides the
ability to create tabular output a single cell at a time while still providing formatting capabilities. These terms frequently appear in tabular output documentation:

cell
   is a collection of data, text, or images that can span multiple rows or columns.

cell padding
   is the thickness on each of the four sides of the cell.

cell spacing
   is the thickness of the spacing between cells.

dimension unit
   is a nonnegative number followed by an optional unit of measure. A number without a unit of measure is interpreted as pixels. It is not recommended that you use pixels because of adverse dependencies on resolution that can differ between destinations.

column
   is the vertical dimension of a 2-D arrangement (a collection of cells in the vertical dimension).

row
   is the horizontal dimension of a 2-D arrangement (a collection of cells in the horizontal dimension).

table
   is a 2-D arrangement of information that is traditionally displayed so that similar categories or classes of information are displayed in a structured manner (a collection of rows).

The following graphic illustrates the table sections that the RWI can create and modify:

The following examples use table and table section methods:

- “Example 7: Creating a Simple Table” on page 216
- “Example 8: Formatting Cells Using the Textdecoration Style Attribute ” on page 219
- “Example 9: Creating a Table with Row and Column Spanning” on page 223
Text Methods

Text methods enable you to produce high-quality formatted text. The text-formatting features can be used whether your text is static or being provided from a data source.

- “Example 1: Formatting Data with Style Attributes and Style Elements” on page 195
- “Example 4: Creating a Flyer with Text Methods” on page 206
- “Example 6: Working with BY Groups and IF-THEN Processing” on page 213

The following methods are text methods:

- “FORMAT_TEXT Method” on page 149
- “NOTE Method” on page 173

ODS Output Object Creation

The Basics

To use an ODS object in your SAS program, you must declare and create (instantiate) the object. The DATA step component interface provides a mechanism for accessing predefined component objects from within the DATA step. For more information about the predefined DATA step component objects, see “Using DATA Step Component Objects” in SAS Language Reference: Concepts.
Declaring and Instantiating an ODS Output Object

The ODS Report Writing Interface uses the DECLARE statement to declare and instantiate an ODS Output Object. For example, in the following line of code, the DECLARE statement declares and instantiates an ODS object and assigns it to the object reference obj:

```plaintext
declare odsout obj();
```

The local variable (obj) communicates with the ODS system and remains active throughout the entire DATA step. As the body of the DATA step is executed once for each input observation during the execution phase, the local object variable (obj) needs to be initialized only once for the duration of the DATA step. This can be done using the `_N_` as follows.

```plaintext
if _N_ = 1 then do;
   declare odsout obj();
end;
```

The following two-step process is an alternative way of declaring and instantiating an ODS object.

```plaintext
declare odsout obj;
obj = _new_ (odsout);
```

The DECLARE statement without ( ) specified creates a local variable for use in the DATA step, but it remains undefined until the `obj = _new_ odsout;` statement is executed. After you declare the new ODS object with the DECLARE statement, use the `_NEW_` operator to instantiate the object.
Chapter 11
Reference and Examples

ODS Output Object Creation .................................................. 122
The Basics ........................................................................ 122
Declaring and Instantiating an ODS Output Object .................. 122

Types of Methods ............................................................... 123
Overview of Report Writing Methods ................................. 123
General Purpose Methods .................................................. 123
Layout Methods .................................................................. 123
Media Methods .................................................................. 125
Page Methods ..................................................................... 126
Table Methods .................................................................... 126
Text Methods ...................................................................... 128

Statements and Methods by Category .................................. 128

Dictionary ............................................................................ 130
DECLARE Statement, ODS Object ....................................... 130
.NEW_. Operator, ODS Object ........................................... 131
AUDIO Method ................................................................... 132
BODY_START Method ...................................................... 135
BODY_END Method .......................................................... 136
CELL_START Method ......................................................... 136
CELL_END Method ............................................................ 141
DELETE Method ................................................................. 142
FOOT_START Method ......................................................... 142
FOOT_END Method ............................................................ 143
FORMAT_CELL Method .................................................... 144
FORMAT_TEXT Method ...................................................... 149
FOOTNOTE Method .......................................................... 152
HEAD_START Method ....................................................... 155
HEAD_END Method ........................................................... 156
HREF Method .................................................................... 157
IMAGE Method ................................................................. 160
LAYOUT_ABSOLUTE Method .............................................. 161
LAYOUT_GRIDDED Method ................................................. 165
LAYOUT_END Method ........................................................ 171
LINE Method ..................................................................... 171
NOTE Method .................................................................... 173
PAGE Method .................................................................... 176
REGION Method, Absolute .................................................. 176
REGION Method, Gridded ................................................... 180
ROW_END Method ............................................................ 184
ROW_START Method .......................................................... 185
ODS Output Object Creation

The Basics

To use an ODS object in your SAS program, you must declare and create (instantiate) the object. The DATA step component interface provides a mechanism for accessing predefined component objects from within the DATA step. For more information about the predefined DATA step component objects, see “Using DATA Step Component Objects” in SAS Language Reference: Concepts.

Declaring and Instantiating an ODS Output Object

The ODS Report Writing Interface uses the DECLARE statement to declare and instantiate an ODS Output Object. For example, in the following line of code, the DECLARE statement declares and instantiates an ODS object and assigns it to the object reference obj:

```sas
declare odsout obj();
```

The local variable (obj) communicates with the ODS system and remains active throughout the entire DATA step. As the body of the DATA step is executed once for each input observation during the execution phase, the local object variable (obj) needs to be initialized only once for the duration of the DATA step. This can be done using _N_ as follows.

```sas
if _N_ = 1 then do;
    declare odsout obj();
end;
```

The following two-step process is an alternative way of declaring and instantiating an ODS object.

```sas
declare odsout obj;
obj = _new_ (odsout);
```
The DECLARE statement without ( ) specified creates a local variable for use in the DATA step, but it remains undefined until the `obj = _new_ odsout;` statement is executed. After you declare the new ODS object with the DECLARE statement, use the `_NEW_` operator to instantiate the object.

---

Types of Methods

Overview of Report Writing Methods

There are various methods that make up the ODS Report Writing Interface. They can be organized into the following types:

- general purpose methods
- layout methods
- media methods
- page methods
- table methods
- text methods

CAUTION:
Not all destinations support the same set of features, and therefore some methods might have restrictions in some destinations.

General Purpose Methods

General purpose methods can be used to create URLs, insert images and lines into your output, or delete class instances.

The following are general purpose methods:

- “DELETE Method” on page 142
- “HREF Method” on page 157
- “LINE Method” on page 171

Layout Methods

Concepts

Layout methods control the placement of output on a page. There are two groups of layout methods, absolute and gridded. Layout containers can be nested, and gridded and absolute layouts can be used together. These terms frequently appear in layout documentation:

- dimension unit

  is a nonnegative number followed by an optional unit of measure. A number without a unit of measure is interpreted as pixels. It is not recommended that you size things in pixels because of adverse dependencies on resolution that can differ between destinations.
footnote
is descriptive text that appears at the bottom of your output. Title and footnote processing is always done before any output is produced on the physical page. This is often referred to as page initialization time. Footnotes can be specified inside regions, but only the footnotes that have changed will be displayed.

layout container
is an area that contains a collection of regions. Layouts can contain only regions. Layouts can have a fixed size (like width=3in and height=4in) or can be dynamically sized to accommodate the regions.

region container
is an area that contains output (like text, tables, images), or nested layout containers. Regions can also have a fixed size, or can be dynamically sized to accommodate the collection of output. Regions cannot contain titles.

title
is descriptive text that appears at the top of your output. Title and footnote processing is always done before any output is produced on the physical page. This is often referred to as page initialization time.
**Gridded Layout Methods**
Gridded layout methods enable you to arrange output in a two-dimensional gridded structure, such as a spreadsheet or piece of graph paper. Gridded layout is a simple, powerful tool for arranging output and managing output on one or more pages. Gridded layouts enable you to do the following:

- enforce automatic alignment of respective grid cells
- continue onto the next page if necessary
- dynamically compute the size of a grid cell

The following are gridded layout methods:

- “LAYOUT_GRIDDED Method” on page 165
- “LAYOUT_END Method” on page 171
- “REGION Method, Gridded” on page 180

**Absolute Layout Methods**
Absolute layout methods enable you to specify the exact location of a layout and region container on a page. This method is very precise. Each container must be explicitly placed to ensure no overlap. Absolute layouts are restricted to a single page. If the output is too large to fit in the fixed-size container, the output is discarded, and you receive a blank region and a warning in your log. Absolute layouts are suitable for static types of output, as in the following situations:

- placing output in a specific location on a pre-printed form
- creating cover pages
- precisely placing output in a nested region container

Absolute layouts are valid in printer family destinations only.

The following example uses absolute layout methods:

- “Example 12: Creating a Cover Page with the LAYOUT_ABSOLUTE Method” on page 243

The following are absolute layout methods:

- “LAYOUT_ABSOLUTE Method” on page 161
- “LAYOUT_END Method” on page 171
- “REGION Method, Absolute” on page 176

**Media Methods**
Media methods create audio and video files for HTML5 or EPUB output, or insert images into your output.

- “VIDEO Method” on page 191
- “AUDIO Method” on page 132
- “IMAGE Method” on page 160
Page Methods

Page methods control characteristics of the entire page. You can start a new page or modify an existing title or footnote. The TITLE and FOOTNOTE methods enable you to incorporate a particular data value into your title or footnote. These methods work similarly to the global TITLE and FOOTNOTE statements available outside of the DATA step.

The following example uses the PAGE method:

- “Example 10: Using the Page Method” on page 228

The following are page methods:

- “PAGE Method” on page 176
- “TITLE Method” on page 188
- “FOOTNOTE Method” on page 152

Table Methods

Table methods create tabular output that consists of row headings, column headings, and data values in a structured format. The ODS Report Writing Interface provides the ability to create tabular output a single cell at a time while still providing formatting capabilities. These terms frequently appear in tabular output documentation:

cell
is a collection of data, text, or images that can span multiple rows or columns.

cell padding
is the thickness on each of the four sides of the cell.

cell spacing
is the thickness of the spacing between cells.

dimension unit
is a nonnegative number followed by an optional unit of measure. A number without a unit of measure is interpreted as pixels. It is not recommended that you use pixels because of adverse dependencies on resolution that can differ between destinations.

column
is the vertical dimension of a 2-D arrangement (a collection of cells in the vertical dimension).

row
is the horizontal dimension of a 2-D arrangement (a collection of cells in the horizontal dimension).

table
is a 2-D arrangement of information that is traditionally displayed so that similar categories or classes of information are displayed in a structured manner (a collection of rows).
The following graphic illustrates the table sections that the RWI can create and modify:

The following examples use table and table section methods:

- “Example 7: Creating a Simple Table” on page 216
- “Example 8: Formatting Cells Using the Textdecoration Style Attribute” on page 219
- “Example 9: Creating a Table with Row and Column Spanning” on page 223
- “Example 10: Using the Page Method” on page 228

The following methods are table section methods:

- “BODY_START Method” on page 135
- “BODY_END Method” on page 136
- “FOOT_START Method” on page 142
- “FOOT_END Method” on page 143
- “HEAD_START Method” on page 155
- “HEAD_END Method” on page 156

The following methods are table methods:

- “TABLE_START Method” on page 187
- “TABLE_END Method” on page 186
- “ROW_START Method” on page 185
- “ROW_END Method” on page 184
- “CELL_START Method” on page 136
- “CELL_END Method” on page 141
- “FORMAT_CELL Method” on page 144
Text Methods

Text methods enable you to produce high-quality formatted text. The text-formatting features can be used whether your text is static or being provided from a data source.

- “Example 1: Formatting Data with Style Attributes and Style Elements” on page 195
- “Example 4: Creating a Flyer with Text Methods” on page 206
- “Example 6: Working with BY Groups and IF-THEN Processing” on page 213

The following methods are text methods:

- “FORMAT_TEXT Method” on page 149
- “NOTE Method” on page 173

Statements and Methods by Category

<table>
<thead>
<tr>
<th>Category</th>
<th>Language Elements</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>General Purpose</td>
<td>DELETE Method (p. 142)</td>
<td>Deletes an instance of a class.</td>
</tr>
<tr>
<td></td>
<td>HREF Method (p. 157)</td>
<td>Creates a link.</td>
</tr>
<tr>
<td></td>
<td>IMAGE Method (p. 160)</td>
<td>Inserts an image into all open output destinations.</td>
</tr>
<tr>
<td></td>
<td>LINE Method (p. 171)</td>
<td>Inserts a horizontal rule (line) across the page.</td>
</tr>
<tr>
<td>Layout: Absolute</td>
<td>LAYOUT_ABSOLUTE Method (p. 161)</td>
<td>Creates an absolute layout container. The LAYOUT_ABSOLUTE method enables you to specify the exact location on a page for your regions and is limited to one page of layout.</td>
</tr>
<tr>
<td></td>
<td>LAYOUT_END Method (p. 171)</td>
<td>Ends a gridded or absolute layout block.</td>
</tr>
<tr>
<td></td>
<td>REGION Method, Absolute (p. 176)</td>
<td>Creates a region container for absolute layouts.</td>
</tr>
<tr>
<td>Layout: Gridded</td>
<td>LAYOUT_GRIDDED Method (p. 165)</td>
<td>Creates a gridded layout container. The LAYOUT_GRIDDED method enables you to arrange output in a two-dimensional gridded structure such as a spreadsheet or a piece of graph paper.</td>
</tr>
<tr>
<td></td>
<td>LAYOUT_END Method (p. 171)</td>
<td>Ends a gridded or absolute layout block.</td>
</tr>
<tr>
<td></td>
<td>REGION Method, Gridded (p. 180)</td>
<td>Creates a region container for gridded layouts.</td>
</tr>
<tr>
<td>Media</td>
<td>AUDIO Method (p. 132)</td>
<td>Embeds audio into HTML5 or EPUB output.</td>
</tr>
<tr>
<td></td>
<td>VIDEO Method (p. 191)</td>
<td>Embeds video into HTML5 or EPUB output.</td>
</tr>
<tr>
<td>Category</td>
<td>Language Elements</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>--------------------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Page</td>
<td>FOOTNOTE Method (p. 152)</td>
<td>Adds a footnote to the page.</td>
</tr>
<tr>
<td></td>
<td>PAGE Method (p. 176)</td>
<td>Creates a page break.</td>
</tr>
<tr>
<td></td>
<td>TITLE Method (p. 188)</td>
<td>Adds a title to a page.</td>
</tr>
<tr>
<td>Statement</td>
<td>DECLARE Statement, ODS Object (p. 130)</td>
<td>Declares an ODS object and creates an instance of and initializes data for an ODS object.</td>
</tr>
<tr>
<td></td>
<td><em>NEW</em> Operator, ODS Object (p. 131)</td>
<td>Creates an instance of an ODS object.</td>
</tr>
<tr>
<td>Table</td>
<td>BODY_START Method (p. 135)</td>
<td>Specifies the start of the body section of the table.</td>
</tr>
<tr>
<td></td>
<td>BODY_END Method (p. 136)</td>
<td>Specifies the end of the body section of the table.</td>
</tr>
<tr>
<td></td>
<td>CELL_START Method (p. 136)</td>
<td>Specifies the start of a cell.</td>
</tr>
<tr>
<td></td>
<td>CELL_END Method (p. 141)</td>
<td>Specifies the end of a cell.</td>
</tr>
<tr>
<td></td>
<td>FOOT_START Method (p. 142)</td>
<td>Specifies the start of the footer section of a table.</td>
</tr>
<tr>
<td></td>
<td>FOOT_END Method (p. 143)</td>
<td>Specifies the end of the footer section of a table.</td>
</tr>
<tr>
<td></td>
<td>FORMAT_CELL Method (p. 144)</td>
<td>Formats the contents of a cell.</td>
</tr>
<tr>
<td></td>
<td>HEAD_START Method (p. 155)</td>
<td>Specifies the start of the header section of a table.</td>
</tr>
<tr>
<td></td>
<td>HEAD_END Method (p. 156)</td>
<td>Specifies the end of the header section of a table.</td>
</tr>
<tr>
<td></td>
<td>ROW_END Method (p. 184)</td>
<td>Specifies the end of a row.</td>
</tr>
<tr>
<td></td>
<td>ROW_START Method (p. 185)</td>
<td>Specifies the beginning of a row.</td>
</tr>
<tr>
<td></td>
<td>TABLE_END Method (p. 186)</td>
<td>Specifies the end of a table.</td>
</tr>
<tr>
<td></td>
<td>TABLE_START Method (p. 187)</td>
<td>Specifies the start of a table.</td>
</tr>
<tr>
<td>Text</td>
<td>FORMAT_TEXT Method (p. 149)</td>
<td>Writes text to the active output destination(s).</td>
</tr>
<tr>
<td></td>
<td>NOTE Method (p. 173)</td>
<td>Writes a note to the active output destination(s).</td>
</tr>
</tbody>
</table>
Dictionary

DECLARE Statement, ODS Object

Declares an ODS object and creates an instance of and initializes data for an ODS object.

Valid in: DATA step
Category: Statement
Alias: DCL
Example: declare odsout obj();
Examples: “Example 1: Formatting Data with Style Attributes and Style Elements” on page 195
“Example 4: Creating a Flyer with Text Methods” on page 206
“Example 5: Working with Data Sets” on page 210
“Example 6: Working with BY Groups and IF-THEN Processing” on page 213
“Example 7: Creating a Simple Table” on page 216
“Example 8: Formatting Cells Using the Textdecoration Style Attribute” on page 219
“Example 9: Creating a Table with Row and Column Spanning” on page 223
“Example 10: Using the Page Method” on page 228
“Example 12: Creating a Cover Page with the LAYOUT_ABSOLUTE Method” on page 243

Syntax

DECLARE ODSOUT object-reference <();

Required Arguments

ODSOUT
is the class name.

object-reference
specifies the object reference name for the ODS object.

<table>
<thead>
<tr>
<th>Length</th>
<th>1-32</th>
</tr>
</thead>
</table>
| Restrictions | object-reference must start with a letter or an underscore (_) followed by a combination of letters, numbers, and underscores
| | object-reference cannot contain blanks or special characters except for an underscore.
| | The maximum length of the object-reference is 32 characters

() instantiates the ODS object. As an alternative to the two-step process of using the DECLARE statement and the _NEW_ operator to declare and instantiate an ODS object, you can use the DECLARE statement to declare and instantiate the ODS object in one step. For example, in the following line of code, the DECLARE
statement declares and instantiates an ODS object and assigns it to the object reference obj:

```sas
declare odsout obj();
```

**Details**

**The Basics**
To use an ODS object in your SAS program, you must declare and create (instantiate) the object. The DATA step component interface provides a mechanism for accessing predefined component objects from within the DATA step. For more information about the predefined DATA step component objects, see “Using DATA Step Component Objects” in *SAS Language Reference: Concepts*.

**Declaring and Instantiating an ODS Output Object**
The ODS Report Writing Interface uses the DECLARE statement to declare and instantiate an ODS Output Object. For example, in the following line of code, the DECLARE statement declares and instantiates an ODS object and assigns it to the object reference obj:

```sas
declare odsout obj();
```

The local variable (obj) communicates with the ODS system and remains active throughout the entire DATA step. As the body of the DATA step is executed once for each input observation during the execution phase, the local object variable (obj) needs to be initialized only once for the duration of the DATA step. This can be done using `_N_` as follows.

```sas
if _N_ = 1 then do;
  declare odsout obj();
end;
```

The following two-step process is an alternative way of declaring and instantiating an ODS object.

```sas
declare odsout obj;
obj= _new_ (odsout);
```

The DECLARE statement without ( ) specified creates a local variable for use in the DATA step, but it remains undefined until the `obj = _new_ odsout;` statement is executed. After you declare the new ODS object with the DECLARE statement, use the `NEW_` operator to instantiate the object.

---

**_NEW_ Operator, ODS Object**
Creates an instance of an ODS object.

**Valid in:** DATA step  
**Category:** Statement  
**Applies to:** ODS object

**Syntax**

```
object-reference= _NEW_ (ODSOUT);
```
**Required Arguments**

*object-reference*

specifies the object reference name for the ODS object.

<table>
<thead>
<tr>
<th>Length</th>
<th>1-32</th>
</tr>
</thead>
</table>

**Restrictions**

*object-reference* must start with a letter or an underscore (_) followed by a combination of letters, numbers, and underscores.

*object-reference* cannot contain blanks or special characters except for an underscore.

The maximum length of the *object-reference* is 32 characters.

**ODSOUT**

is the class name.

**Comparisons**

As an alternative to the two-step process of using the DECLARE statement and the _NEW_ operator to declare and instantiate an ODS object, you can use the DECLARE statement to declare and instantiate the ODS object in one step. For example, in the following line of code, the DECLARE statement declares and instantiates an ODS object and assigns it to the object reference obj:

```
declare odsout obj();
```

**AUDIO Method**

Embeds audio into HTML5 or EPUB output.

- **Valid in:** DATA step
- **Category:** Media
- **Restriction:** The AUDIO method can only be used with the ODS HTML5, ODS EPUB, and ODS EPUB3 statements.

**Example:**

```sas
ods html5 file="audio.html";

data _null_
  dcl odsout obj();
  obj.audio( file: "filepath-here.mp3", type: "mp3", preload: "auto", autoplay: "off", loop: "no" );
run;
ods html5 close;
```
Example: For a complete example using the Audio method, see "Example 2: Adding Audio to HTML5 Output" on page 200.

Syntax

```
AUDIO (FILE:"path" | "FILEREF:fileref"
< TYPE:"audio-type">,<audio-control-option-1>,<audio-control-option-2,...>);
```

Required Argument

**FILE: "path" | "FILEREF:fileref"**

specifies the path of the audio to insert into HTML5 or EPUB output.

- **path** is the path to an external audio file. *path* can be a URL or a filename with a fully qualified path.

  Requirement *path* must be enclosed in quotation marks.

- **FILEREF:fileref** is a file reference that has been assigned to an external file.

  To use a FILEREF with the AUDIO method, you must use the URL option in the FILENAME statement, and then specify the fileref with the FILEREF: identifier.

  Requirement FILEREF:fileref must be enclosed in quotation marks.

See

For information about using the URL access method, see “FILENAME Statement, URL Access Method” in *SAS Statements: Reference*.

Example

```
filename AudFile url "url-path.mp3";
obj.audio(file:"fileref:AudFile", type:mp3);
```

Tips

To display multiple audio files, you must use separate AUDIO methods.

- To specify multiple format of the same audio file, you can use multiple FILE: arguments. The browser will then play the first file in the list that is a type that it supports. This enables you to support multiple browsers with one AUDIO method.

Optional Arguments

**TYPE: "audio-type"**

specifies the audio type. If the TYPE: option is not specified, then the browser will determine the file type. All other options will then default to browser-determined values.

Most browsers support one or more of the following audio types:

- MP3
- OGG
- WAV
Table 11.1  Supported Audio Types for Browsers

<table>
<thead>
<tr>
<th>Browser</th>
<th>Audio Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Internet Explorer</td>
<td>MP3</td>
</tr>
<tr>
<td>Chrome</td>
<td>MP3, Wav, Ogg</td>
</tr>
<tr>
<td>Firefox</td>
<td>MP3, Wav, Ogg</td>
</tr>
<tr>
<td>Safari</td>
<td>MP3, Wav</td>
</tr>
<tr>
<td>Opera</td>
<td>MP3, Wav, Ogg</td>
</tr>
</tbody>
</table>

Requirement  If TYPE is specified, it must be specified before the next FILE option is specified.

PRELOAD: "AUTO | NONE | METADATA"
specifies how the audio file loads in a browser.

AUTO  load the audio file as soon as the page loads
NONE  load the audio file when user presses the play button
METADATA  load information about the audio file when the page loads, but not the audio itself.

Default  AUTO

AUTOPLAY: "OFF | ON"
specifies when the audio begins playing.

OFF  start playing the audio when the play button is pushed.

Alias  NO

ON  start playing the audio immediately when the page loads.

Alias  YES

Default  OFF

LOOP:"OFF | ON"
specifies whether the audio is continuously replayed.

OFF  play the audio once.

Alias  NO

ON  start replaying the audio immediately after it ends.

Alias  YES
MUTED: "OFF | ON"
controls whether the audio is muted.

OFF
the audio is on when the file starts playing.

ON
the audio is off when the file starts playing.

Default OFF

Restriction The MUTED: option is not supported in Internet Explorer 9 or earlier.

---

**BODY_START Method**

Specifies the start of the body section of the table.

Valid in: DATA step
Category: Table
Requirement: The BODY_START method must be used with the BODY_END method.
Tip: Specifying the TYPE argument in the ROW_START method is an alternate approach to using the BODY_START method.
See: For information about terminology and concepts used to create tables, see “Terminology” on page 112.

Example:

```javascript
obj.table_start();
obj.body_start();
  obj.row_start();
    obj.format_cell(data: 'A single cell table');
  obj.row_end();
obj.body_end();
obj.table_end();
```

**Syntax**

`BODY_START(<option-1><option-2, ...>);`

**Optional Arguments**

**STYLE_ATTR**: `<character-variable | 'style-attribute'>`
specifies the style attributes to override those defined in the selected style element. The specified style attributes then apply to all of the text specified in the method, no matter where STYLE_ATTR is specified within the method.

`character-variable` specifies the name of a variable from the input data set.
**STYLE_ELEM:** `<character-variable | 'style-element'>`

Specifies the style element that contains the collection of style attributes to be applied to the data value. The style element then applies to all of the text specified in the method, no matter where STYLE_ELEM is specified within the method.

- **character-variable** specifies the name of a variable from the input data set.
- **style-element** specifies a style element.

**Default**
The default style element for the BODY_START methods is Body.

**Requirement**
You must enclose `style-element` in quotation marks.

**See**
For a list of style elements, see “Style Elements” in SAS Output Delivery System: User's Guide.

### BODY_END Method

Specifies the end of the body section of the table.

- **Valid in:** DATA step
- **Category:** Table
- **Requirement:** The BODY_END method must be used with the BODY_START method.
- **See:** For information about terminology and concepts used to create tables, see “Terminology” on page 112.

**Example:**
```javascript
obj.table_start();
obj.body_start();
    obj.row_start();
    obj.format_cell(data: 'A single cell table');
    obj.row_end();
    obj.body_end();
obj.table_end();
```

**Syntax**

```javascript
BODY_END();
```

### CELL_START Method

Specifies the start of a cell.

- **Valid in:** DATA step
Category: Table

Requirement: The CELL_START method must be used with the CELL_END method.

Tip: You can use the CELL_START method instead of the FORMAT_CELL method if you want to populate a cell with nested tables, layouts, or the FORMAT_TEXT method.

See: For information about terminology and concepts used to create tables, see “Terminology” on page 112.

Example:
```plaintext
obj.table_start();
    obj.row_start();
        obj.cell_start();
            obj.format_text(data: 'A single cell table');
        obj.cell_end();
    obj.row_end();
obj.table_end();
```

Syntax

```plaintext
CELL_START(<option-1>,<option-2, ...>);
```

Optional Arguments

**COLUMN_SPAN:** number | numeric-variable

specifies the number of grid columns that the cell will occupy. The COLUMN_SPAN argument enables you to combine adjacent cells.

- **number** is a positive integer.
- **numeric-variable** is the name of a numeric variable from the input data set.

Default 1

Example
```plaintext
exmpl.cell_start(column_span: 3);
```

**DATA:** <"string" | number | character-variable | numeric-variable>

specifies the data to display.

- **'string'** specifies a text string.
- **number** specifies an integer.
- **character-variable** specifies the name of a character variable. The value of the character variable from the input data set is written to the output.
- **numeric-variable** specifies the name of a numeric variable. The value of the numeric variable from the input data set is written to the output.

Requirement You must enclose string in quotation marks.

Example
```plaintext
obj.cell_start(data: 'Specifies Text.');
```

**FORMAT:** <character-variable | 'string'>

specifies the format to be applied to the data argument.
character-variable specifies the name of a character variable. The value of the character variable from the input data set is written to the output.

string-variable specifies the name of a user-defined format or a format supplied by SAS.

Default If the data is numeric and no format has been specified, then the data value is formatted using the BEST. format.

Requirement You must enclose string in quotation marks.

Example `exmpl.cell_start(data: sex, column_span: 4, format: '$gender.');`

**HEIGHT: 'dimension'**
specifies the vertical width of the cell.

dimension is a nonnegative number followed by an optional unit of measure. It is not recommended that you use pixels because of adverse dependencies on resolution that can differ between destinations.

The following units of measure are valid:

- cm centimeters
- em standard typesetting measurements unit for width
- ex standard typesetting measurements unit for height
- in inches
- mm millimeters
- px pixels
- pt a printer’s point

Default If you omit the HEIGHT argument, the default is the maximum vertical space needed to display the output contained in the cell.

Restriction The height is restricted by the dimensions of the cell.

Requirement *dimension* must be enclosed in quotation marks.

**INHIBIT: ('B' | 'L' | 'R' | 'T' | 'X')**
specifies the sides of the cell that will not be displayed.

Valid values:

- B specifies not to draw the bottom border. Specifying the B value is ineffective if the destination already drew the rule on the top of the previous cell.
- L specifies not to draw the left border. Specifying the L value is ineffective if the destination already drew that rule on the right of the previous cell.
- R specifies not to draw the right border.
- T specifies not to draw the top border. Specifying the T value is ineffective if the destination already drew a rule at the bottom of the previous row.
X specifies not to draw the contents of the cell, only the background of the cell. Specifying the X value on one of two cells that are using the B or R values is helpful.

Restrictions The INHIBIT argument is not supported by the HTML destination. The values for the INHIBIT argument must be enclosed in quotation marks.

**INLINE_ATTR:** `<character-variable | 'style-attribute'>`

specifies the style attributes to override those defined in the selected style element. The style attributes specified by the INLINE_ATTR argument apply only to the preceding DATA argument.

*character-variable* specifies the name of a variable from the input data set.

*style-attribute* specifies a style attribute.

**Requirement** You must enclose *style-attribute* in quotation marks.

**See** For a list of style attributes, see “Style Attributes” in *SAS Output Delivery System: User’s Guide*.

**Example**
```sas
exmpl.cell_start(data: 'Make this text red',
                 inline_attr: 'color=red',
                 data: 'and this bold',
                 inline_attr: 'font_weight=bold',
                 data: 'and use a 16pt font',
                 inline_attr: 'font_size=16pt');
```

**INLINE_ELEM:** `<character-variable | 'style-element'>`

specifies the style element that contains the collection of style attributes to be applied to the data value. The style element specified by the INLINE_ELEM argument applies only to the preceding DATA argument.

*character-variable* specifies the name of a variable from the input data set.

*'style-element'* specifies a style element.

**Default** The default style element for the CELL_START method is Text.

**Requirement** You must enclose *style-element* in quotation marks.

**See** For a list of style elements, see “Style Elements” in *SAS Output Delivery System: User’s Guide*.

**Example**
```sas
exmpl.cell_start(data: 'This text is formatted using the INLINE_ELEM argument and the Output Style Element.',
                 inline_elem: 'Output',
                 data: 'The INLINE_ELEM argument applies only to the preceding text.');
```

**JUST:** ‘C’ | ‘L’ | ‘R’ | char-variable

specifies the horizontal justification for the data value.

*C* specifies center justification.

*L* specifies left justification.

**Restrictions** The INHIBIT argument is not supported by the HTML destination. The values for the INHIBIT argument must be enclosed in quotation marks.
R specifies right justification.  

(char-attribute) specifies the name of a character variable from the input data set.  

Default C  

Requirement You must enclose C, L, and R in quotation marks.  

**ROW_SPAN:** number | numeric variable  

specifies the number of rows that the cell will occupy. The ROW_SPAN argument enables you to combine adjacent cells.  

number is a positive integer.  

numeric-variable is the name of a numeric variable from the input data set.  

Default 1  

Example `exmpl.cell_start(row_span: 3);`  

**SPLIT:** <character-variable | 'string'>  

specifies the split character to be applied to the data value. A new line will be started when it reaches the specified split character, and will continue on the next line. The split character itself is not part of the data value.  

character-variable is the name of a character variable from the input data set.  

string is one or more characters.  

**STYLE_ATTR:** <character-variable | 'style-attribute'>  

specifies the style attributes to override those defined in the selected style element. The specified style attributes then apply to all of the text specified in the method, no matter where STYLE_ATTR is specified within the method.  

character-variable specifies the name of a variable from the input data set.  

style-attribute specifies a style attribute.  

Requirement You must enclose style-attribute in quotation marks.  

See For a list of style attributes, see “Style Attributes” in *SAS Output Delivery System: User's Guide*.  

**STYLE_ELEM:** <character-variable | 'style-element'>  

specifies the style element that contains the collection of style attributes to be applied to the data value. The style element then applies to all of the text specified in the method, no matter where STYLE_ELEM is specified within the method.  

character-variable specifies the name of a variable from the input data set.  

style-element specifies a style element.  

Default The default style element for the CELL_START method is Text.  

Requirement You must enclose style-element in quotation marks.
VJUST: `<B> | `M` | `T` | character-variable>`
specifies the vertical justification for the data value.

- B specifies bottom justification
- M specifies middle justification
- T specifies top justification
- char-variable specifies the name of a character variable from the input data set.

Default `’T’`

Requirement You must enclose B, M, and T in quotation marks.

URL: `'URL-path'`
specifies the fully qualified URL path.

Example

```
obj.cell_start(data: "Link to SAS Support",
               url: "http://support.sas.com");
```

WIDTH: `’dimension’`
specifies the horizontal width of the cell.

- `dimension` is a nonnegative number followed by an optional unit of measure. It is not recommended that you use pixels because of adverse dependencies on resolution that can differ between destinations.

The following units of measure are valid:

- cm centimeters
- em standard typesetting measurements unit for width
- ex standard typesetting measurements unit for height
- in inches
- mm millimeters
- px pixels
- pt a printer’s point

Default If you omit the WIDTH argument, the default is the maximum horizontal space needed to display the output contained in the cell.

Restriction The width is restricted by the dimensions of the cell.

Requirement `dimension` must be enclosed in quotation marks.
Valid in: DATA step
Category: Table
Requirement: The CELL_END method must be used with the CELL_START method.
See: For information about terminology and concepts used to create tables, see “Terminology” on page 112.
Example:
```javascript
obj.table_start();
  obj.row_start();
    obj.cell_start();
      obj.format_text(data: 'A single cell table');
    obj.cell_end();
  obj.row_end();
obj.table_end();
```

Syntax

CELL_END();

DELETE Method

Deletes an instance of a class.
Valid in: DATA step
Category: General Purpose

Syntax

DELETE();

FOOT_START Method

Specifies the start of the footer section of a table.
Valid in: DATA step
Category: Table
Requirement: The FOOT_START method must be used with the FOOT_END method.
Tip: Specifying the TYPE argument with the ROW_START method is an alternate approach to using the FOOT_START method.
See: For information about terminology and concepts used to create tables, see “Terminology” on page 112.
Example:
```javascript
obj.table_start();
  obj.foot_start();
    obj.row_start();
      obj.cell_start();
        obj.format_text(data: 'A single cell table');
      obj.cell_end();
    obj.row_end();
  obj.table_end();
```
FOOT_END Method

Specifies the end of the footer section of a table.

Valid in: DATA step
Category: Table
Requirement: The FOOT_END method must be used with the FOOT_START method.
See: For information about terminology and concepts used to create tables, see “Terminology” on page 112.

Example:

```java
obj.table_start();
obj.foot_start();
obj.row_start();
obj.cell_start();
```

Syntax

**FOOT_START**(<option-1>,<option-2,...>);

Optional Arguments

**STYLE_ATTR:** `<character-variable> | 'style-attribute'>`
specifies the style attributes to override those defined in the selected style element. The specified style attributes then apply to all of the text specified in the method, no matter where STYLE_ATTR is specified within the method.

- `character-variable` specifies the name of a variable from the input data set.
- `style-attribute` specifies a style attribute.

Requirement: You must enclose `style-attribute` in quotation marks.

See: For a list of style attributes, see “Style Attributes” in *SAS Output Delivery System: User's Guide*.

**STYLE_ELEM:** `<character-variable> | 'style-element'>`
specifies the style element that contains the collection of style attributes to be applied to the data value. The style element then applies to all of the text specified in the method, no matter where STYLE_ELEM is specified within the method.

- `character-variable` specifies the name of a variable from the input data set.
- `style-element` specifies a style element.

Default: The default style element for the FOOT_START method is Footer.

Requirement: You must enclose `style-element` in quotation marks.

Syntax

**FOOT_END( );**

---

**FORMAT_CELL Method**

Formats the contents of a cell.

- **Valid in:** DATA step
- **Category:** Table
- **See:** For information about terminology and concepts used to create tables, see "Terminology" on page 112.

**Examples:**

```javascript
obj.table_start();
obj.row_start();
obj.format_cell(data: 'This is a single cell', style_elem: 'Header');
obj.row_end();
obj.table_end();
```

The following example creates a row with two separate cells.

```javascript
obj.table_start();
obj.row_start();
obj.format_cell(data: 'Cell 1');
obj.format_cell(data: 'Cell 2');
obj.row_end();
obj.table_end();
```

**Examples:**

- "Example 7: Creating a Simple Table" on page 216
- "Example 8: Formatting Cells Using the Textdecoration Style Attribute " on page 219
- "Example 9: Creating a Table with Row and Column Spanning" on page 223
- "Example 10: Using the Page Method" on page 228

---

**Syntax**

```javascript
FORMAT_CELL(DATA:'displayed-data' <,option-1><,option-2, ...>);
```

**Required Argument**

- **DATA: <"string" | number | character-variable | numeric-variable>**
  - specifies the data to display.

    - `'string'` specifies a text string.
    - `number` specifies an integer.
**FORMAT_CELL Method**

character-variable specifies the name of a character variable. The value of the character variable from the input data set is written to the output.

numeric-variable specifies the name of a numeric variable. The value of the numeric variable from the input data set is written to the output.

**Requirement**
You must enclose *string* in quotation marks.

**Example**
```javascript
obj.format_cell(data: 'This text is the contents of a cell.');
```

**Optional Arguments**

**COLUMN_SPAN:** *number | numeric-variable*

specifies the number of columns that the cells will occupy. The COLUMN_SPAN argument enables you to combine adjacent cells.

- *number* is a positive integer.
- *numeric-variable* is the name of a numeric variable from the input data set.

**Default**
1

**Example**
```javascript
obj.format_cell(data: 'This is a spanning cell',
    column_span: 2,
    style_elem: 'Header');
```

**FORMAT:** `<character-variable | 'string'>`

specifies the format to be applied to the data argument.

- *character-variable* specifies the name of a character variable. The value of the character variable from the input data set is written to the output.
- *string-variable* specifies the name of a user-defined format or a format supplied by SAS.

**Default**
If the data is numeric and no format has been specified, then the data value is formatted using the BEST. format.

**Requirement**
You must enclose *string* in quotation marks.

**Example**
```javascript
exmpl.format_cell(data: bmi, format: '8.2');
```

**HEIGHT:** *'dimension'*

specifies the vertical width of the cell.

- *dimension* is a nonnegative number followed by an optional unit of measure. It is not recommended that you use pixels because of adverse dependencies on resolution that can differ between destinations.

The following units of measure are valid:

- *cm* centimeters
- *em* standard typesetting measurements unit for width
ex standard typesetting measurements unit for height
in inches
mm millimeters
px pixels
pt a printer’s point

Requirement *dimension* must be enclosed in quotation marks.

**INHIBIT: ('B' | 'L' | 'R' | 'T' | 'X')**
specifies the sides of the cell that will not be displayed.

**Valid values:**

- **B** specifies not to draw the bottom border. Specifying the B value is ineffective if the destination already drew the rule on the top of the previous cell.
- **L** specifies not to draw the left border. Specifying the L value is ineffective if the destination already drew that rule on the right of the previous cell.
- **R** specifies not to draw the right border.
- **T** specifies not to draw the top border. Specifying the T value is ineffective if the destination already drew a rule at the bottom of the previous row.
- **X** specifies not to draw the contents of the cell, only the background of the cell. Specifying the X value on one of two cells that are using the B or R values is helpful.

**Restrictions** The INHIBIT argument is not supported by the HTML destination.

values for the INHIBIT argument must be enclosed in quotation marks.

**INLINE_ATTR:** `<character-variable | 'style-attribute'>`
specifies the style attributes to override those defined in the selected style element. The style attributes specified by the INLINE_ATTR argument apply only to the preceding DATA argument.

- **character-variable** specifies the name of a variable from the input data set.
- **style-attribute** specifies a style attribute.

**Requirement** You must enclose style-attribute in quotation marks.

**See** For a list of style attributes, see “Style Attributes” in SAS Output Delivery System: User's Guide.

**Example**

```sas
exmpl.format_cell (data: 'Make this text red',
                   inline_attr: 'color=red',
                   data: 'and this bold',
                   inline_attr: 'font_weight=bold',
                   data: 'and use a 16pt font',
                   inline_attr: 'font_size=16pt');
```
INLINE_ELEM: `<character-variable | 'style-element'>`

specifies the style element that contains the collection of style attributes to be applied to the data value. The style element specified by the INLINE_ELEM argument applies only to the preceding DATA argument.

- **character-variable** specifies the name of a variable from the input data set.
- **'style-element'** specifies a style element.

**Default**
The default style element for the FORMAT_CELL method is Text.

**Requirement**
You must enclose **style-element** in quotation marks.

**See**
For a list of style elements, see “Style Elements” in *SAS Output Delivery System: User’s Guide*.

**Example**
```sas
exmpl.format_cell(data: 'This text is formatted using the INLINE_ELEM argument and the Output Style Element.', inline_elem: 'Output', data: 'The INLINE_ELEM argument applies only to the preceding text.');
```

JUST: 'C' | 'L' | 'R' | char-variable

specifies the horizontal justification for the data value.

- **C** specifies center justification.
- **L** specifies left justification.
- **R** specifies right justification.
- **char-variable** specifies the name of a character variable from the input data set.

**Default**
C

**Requirement**
You must enclose C, L, and R in quotation marks.

ROW_SPAN: `<number | numeric-variable >`

specifies the number of rows that the cell will occupy. The ROW_SPAN argument enables you to combine adjacent cells in a layout.

- **number** is a positive integer.
- **numeric-variable** is the name of a numeric variable from the input data set.

**Default**
1

**Example**
```sas
exmpl.format_cell(data: 'Name', row_span: 3, vjust: 'B');
```

SPLIT: `<character-variable | 'string'>`

specifies the split character to be applied to the data value. A new line will be started when it reaches the specified split character, and will continue on the next line. The split character itself is not part of the data value.

- **character-variable** is the name of a character variable from the input data set.

The character variable’s value resolves to the character.
string is one or more characters.

**STYLE_ATTR**: `<character-variable | 'style-attribute'>`

specifies the style attributes to override those defined in the selected style element. The specified style attributes then apply to all of the text specified in the method, no matter where STYLE_ATTR is specified within the method.

- **character-variable** specifies the name of a variable from the input data set.
- **style-attribute** specifies a style attribute.

**Requirement** You must enclose **style-attribute** in quotation marks.

**See** For a list of style attributes, see “Style Attributes” in *SAS Output Delivery System: User’s Guide*.

**Example**
```sas
exmpl.format_cell(data: 'With the STYLE_ATTR argument, ',
    style_attr: 'color=red',
    data: 'the style attribute applies to all text.');
```

**STYLE_ELEM**: `<character-variable | 'style-element'>`

specifies the style element that contains the collection of style attributes to be applied to the data value. The style element then applies to all of the text specified in the method, no matter where STYLE_ELEM is specified within the method.

- **character-variable** specifies the name of a variable from the input data set.
- **style-element** specifies a style element.

**Default** The default style element for the FORMATCELL method is Text.

**Requirement** You must enclose **style-element** in quotation marks.

**See** For a list of style elements, see “Style Elements” in *SAS Output Delivery System: User’s Guide*.

**Example**
```sas
exmpl.format_cell(data: 'This text is formatted using STYLE_ELEM and the SystemTitle Style Element.',
    style_elem: 'SystemTitle',
    data: 'The STYLE_ELEM argument applies to the entire method.');
```

**VJUST**: `<B | 'M' | 'T' | character-variable>`

specifies the vertical justification for the data value.

- **B** specifies bottom justification
- **M** specifies middle justification
- **T** specifies top justification
- **character-variable** specifies the name of a character variable from the input data set.

**Default** ‘T’

**Requirement** You must enclose B, M, and T in quotation marks.

**URL**: `'URL-path'`

specifies the fully qualified URL path.
**WIDTH:** `'dimension'`

specifies the horizontal width of the cell.

`dimension`

is a nonnegative number followed by an optional unit of measure. It is not recommended that you use pixels because of adverse dependencies on resolution that can differ between destinations.

The following units of measure are valid:

- **cm** centimeters
- **em** standard typesetting measurements unit for width
- **ex** standard typesetting measurements unit for height
- **in** inches
- **mm** millimeters
- **px** pixels
- **pt** a printer’s point

**Requirement** `dimension` must be enclosed in quotation marks.

---

**FORMAT_TEXT Method**

Writes text to the active output destination(s).

**Valid in:** DATA step

**Category:** Text

**Example:**

```r
obj.format_text(data: strip(street),
    styleAttr: 'font_style=italic
    font_size=12pt
    font_weight=bold
    color=cxbbb2e0');
```

**Examples:**

- “Example 1: Formatting Data with Style Attributes and Style Elements” on page 195
- “Example 4: Creating a Flyer with Text Methods” on page 206
- “Example 5: Working with Data Sets” on page 210
- “Example 6: Working with BY Groups and IF-THEN Processing” on page 213

**Syntax**

```
FORMAT_TEXT(DATA:'displayed-data' <,option-1><,option-2, ...>);
```

**Required Argument**

**DATA:** `<"string" | number | character-variable | numeric-variable`

specifies the data to display.

`'string'` specifies a text string.
number             specifies an integer.

character-variable specifies the name of a character variable. The value of the character variable from the input data set is written to the output.

numeric-variable specifies the name of a numeric variable. The value of the numeric variable from the input data set is written to the output.

**Requirement** You must enclose string in quotation marks.

**Examples**
The following code displays the specified text to active output destinations.
```javascript
obj.format_text(data: 'Display this text to active output destinations.);
```

The following code specifies that the value of the variable Name is displayed to active output destinations.
```javascript
obj.format_text(data: name);
```

---

**Optional Arguments**

**FORMAT**: `<character-variable | string>`
specifies the format to be applied to the data argument.

- **character-variable** specifies the name of a character variable. The value of the character variable from the input data set is written to the output.
- **string-variable** specifies the name of a user-defined format or a format supplied by SAS.

**Default** If the data is numeric and no format has been specified, then the data value is formatted using the BEST. format.

**Requirement** You must enclose string in quotation marks.

**Example**
```javascript
exmpl.format_text(data: bmi, format: '8.2');
```

**INLINE_ATTR**: `<character-variable | style-attribute>`
specifies the style attributes to override those defined in the selected style element. The style attributes specified by the INLINE_ATTR argument apply only to the preceding DATA argument.

- **character-variable** specifies the name of a variable from the input data set.
- **style-attribute** specifies a style attribute.

**Requirement** You must enclose style-attribute in quotation marks.

**See** For a list of style attributes, see “Style Attributes” in *SAS Output Delivery System: User's Guide*.

**Example**
```javascript
exmpl.format_text (data: 'Make this text red',
                     inline_attr: 'color=red',
                     data: 'and this bold',
                     inline_attr: 'font_weight=bold',
                     data: ' and use a 16pt font',
```

---
INLINE_ELEM: `<character-variable | 'style-element'>`
specifies the style element that contains the collection of style attributes to be applied
to the data value. The style element specified by the INLINE_ELEM argument
applies only to the preceding DATA argument.

- `character-variable` specifies the name of a variable from the input data set.
- `'style-element'` specifies a style element.

**Default**
The default style element for the FORMAT_TEXT method is UserText.

**Requirement**
You must enclose `style-element` in quotation marks.

**See**
For a list of style elements, see “Style Elements” in SAS Output Delivery System:
User’s Guide.

**Example**
```javascript
exmpl.format_text(data: 'This text is formatted using the
INLINE_ELEM argument and the Output Style Element.',
inline_elem: 'Output',
data: 'The INLINE_ELEM argument applies
only to the preceding text.');
```

JUST: 'C' | 'L' | 'R' | `char-variable`
specifies the horizontal justification for the data value.

- C specifies center justification.
- L specifies left justification.
- R specifies right justification.
- `char-variable` specifies the name of a character variable from the input data set.

**Default**
L

**Requirement**
You must enclose C, L, and R in quotation marks.

SPLIT: `<character-variable | 'string'>`
specifies the split character to be applied to the data value. A new line will be started
when it reaches the specified split character, and will continue on the next line. The
split character itself is not part of the data value.

- `character-variable` is the name of a character variable from the input data set.
- `string` is one or more characters.

**Example**
```javascript
obj.format_text(data: 'Line 1#Line 2', split: '#');
```

STYLE_ATTR: `<character-variable | 'style-attribute'>`
specifies the style attributes to override those defined in the selected style element.
The specified style attributes then apply to all of the text specified in the method, no
matter where STYLE_ATTR is specified within the method.

- `character-variable` specifies the name of a variable from the input data set.
- `style-attribute` specifies a style attribute.
Requirement: You must enclose *style-attribute* in quotation marks.

See: For a list of style attributes, see “Style Attributes” in *SAS Output Delivery System: User's Guide*.

Example:
```
exmpl.format_text(data: 'With the STYLE_ATTR argument, ',
                  style_attr: 'color=red',
                  data: 'the style attribute applies to all text.');
```

**STYLE_ELEM:** `<character-variable> | 'style-element'>`

specifies the style element that contains the collection of style attributes to be applied to the data value. The style element then applies to all of the text specified in the method, no matter where STYLE_ELEM is specified within the method.

- **character-variable**: specifies the name of a variable from the input data set.
- **style-element**: specifies a style element.

Default: The default style element for the FORMAT_TEXT method is UserText.

Requirement: You must enclose *style-element* in quotation marks.


**VJUST:** `<'B' | 'M' | 'T' | character-variable>`

specifies the vertical justification for the data value.

- **B**: specifies bottom justification
- **M**: specifies middle justification
- **T**: specifies top justification
- **char-variable**: specifies the name of a character variable from the input data set.

Default: ‘T’

Requirement: You must enclose B, M, and T in quotation marks.

**URL:** `'URL-path'`

specifies the fully qualified URL path.

Example:
```
obj.format_text(data: "Link to SAS Support",
                url: "http://support.sas.com");
```

**FOOTNOTE Method**

Adds a footnote to the page.

- **Category:** Page
- **Example:** The following example left-justifies the footnote:
obj.footnote(data: 'Left Justified Footnote',
    style_attr: 'just=left');

Syntax

FOOTNOTE(DATA: 'displayed-data' <,option-1><,option-2, ...>);

Required Argument

DATA: <"string" | number | character-variable | numeric-variable>

specifies the data to display.

'"string" specifies a text string.

number specifies an integer.

character-variable specifies the name of a character variable. The value of the character variable from the input data set is written to the output.

numeric-variable specifies the name of a numeric variable. The value of the numeric variable from the input data set is written to the output.

Requirement You must enclose string in quotation marks.

Example

obj.footnote(data: 'Here is a Footnote.');

Optional Arguments

CLEAR: line-number

clears the footnote on the line number specified.

line-number is the line number.

Range 1–10

Example

The following example clears the previous footnote.

obj.footnote(data: 'Here is a footnote', start: 1, clear: 1);

FORMAT: <character-variable | "string">

specifies the format to be applied to the data argument.

character-variable specifies the name of a character variable. The value of the character variable from the input data set is written to the output.

string-variable specifies the name of a user-defined format or a format supplied by SAS.

Default If the data is numeric and no format has been specified, then the data value is formatted using the BEST. format.

Requirement You must enclose string in quotation marks.

Example

exmpl.footnote(data: date, format: 'date9.');
INLINE_ATTR: `<character-variable | 'style-attribute'>`
specifies the style attributes to override those defined in the selected style element. The style attributes specified by the INLINE_ATTR argument apply only to the preceding DATA argument.

- **character-variable** specifies the name of a variable from the input data set.
- **style-attribute** specifies a style attribute.

**Requirement**
You must enclose style-attribute in quotation marks.

**See**
For a list of style attributes, see “Style Attributes” in SAS Output Delivery System: User's Guide.

**Example**
```sas
exmpl.footnote (data: 'Make this text red',
inline_attr: 'color=red',
data: 'and this bold',
inline_attr: 'font_weight=bold',
data: 'and use a 16pt font',
inline_attr: 'font_size=16pt');
```

INLINE_ELEM: `<character-variable | 'style-element'>`
specifies the style element that contains the collection of style attributes to be applied to the data value. The style element specified by the INLINE_ELEM argument applies only to the preceding DATA argument.

- **character-variable** specifies the name of a variable from the input data set.
- **style-element** specifies a style element.

**Requirement**
You must enclose style-element in quotation marks.

**See**
For a list of style elements, see “Style Elements” in SAS Output Delivery System: User's Guide.

**Example**
```sas
exmpl.footnote (data: 'This text is formatted using the
INLINE_ELEM argument and the Output Style Element.',
inline_elem: 'Output',
data: 'The INLINE_ELEM argument applies only to the preceding text.');
```

SPLIT: `<character-variable | 'string'>`
specifies the split character to be applied to the data value. A new line will be started when it reaches the specified split character, and will continue on the next line. The split character itself is not part of the data value.

- **character-variable** is the name of a character variable from the input data set.
- **string** is one or more characters.

**START: line-number**
specifies the line the footnote will be displayed on.

- **line-number** is the line number.

**Range**
1–10

**Example**
The following example starts the footnote on line number 2.
HEAD_START Method

Specifies the start of the header section of a table.

Valid in: DATA step
Category: Table
Requirement: The HEAD_START method must be used with the HEAD_END method.
Tip: Specifying the TYPE: argument with the ROW_START method is an alternative to using the HEAD_START method.
See: For information about terminology and concepts used to create tables, see “Terminology” on page 112.

Example:
```
obj.table_start();
obj.head_start();
obj.row_start();
```
Examples: “Example 7: Creating a Simple Table” on page 216
“Example 9: Creating a Table with Row and Column Spanning” on page 223

Syntax

HEAD_START(<option-1>,<option-2>, ...>);

Optional Arguments

STYLE_ATTR: <character-variable | 'style-attribute'>
specifies the style attributes to override those defined in the selected style element. The specified style attributes then apply to all of the text specified in the method, no matter where STYLE_ATTR is specified within the method.

character-variable specifies the name of a variable from the input data set.

style-attribute specifies a style attribute.

Requirement You must enclose style-attribute in quotation marks.

See For a list of style attributes, see “Style Attributes” in SAS Output Delivery System: User's Guide.

STYLE_ELEM: <character-variable | 'style-element'>
specifies the style element that contains the collection of style attributes to be applied to the data value. The style element then applies to all of the text specified in the method, no matter where STYLE_ELEM is specified within the method.

character-variable specifies the name of a variable from the input data set.

style-element specifies a style element.

Default The default style element for the HEAD_START method is Header.

Requirement You must enclose style-element in quotation marks.

See For a list of style elements, see “Style Elements” in SAS Output Delivery System: User's Guide.

HEAD_END Method

Specifies the end of the header section of a table.

Valid in: DATA step

Category: Table

Requirement: The HEAD_END method is always used with the HEAD_START method.

See: For information about terminology and concepts used to create tables, see “Terminology” on page 112.
Example:

```javascript
obj.table_start();
obj.head_start();
obj.row_start();
    obj.format_cell(data: 'A single cell table');
obj.row_end();
obj.head_end();
obj.table_end();
```

Examples:

“Example 7: Creating a Simple Table” on page 216
“Example 9: Creating a Table with Row and Column Spanning” on page 223

Syntax

```
HEAD_END();
```

---

HREF Method

Creates a link.

**Valid in:** DATA step

**Category:** General Purpose

**Example:**

```
exmpl.href(data: 'Link to SAS Support',
            url: 'http://support.sas.com',
            format:'$upcase10.');
```

---

Syntax

```
HREF (DATA: 'displayed-data', URL:'data-path' <option-1> <option-2,...>);
```

**Required Arguments**

- **DATA:** `<"string" | number | character-variable | numeric-variable>`
  - specifies the data to display.
    - `'string'` specifies a text string.
    - `number` specifies an integer.
    - `character-variable` specifies the name of a character variable. The value of the character variable from the input data set is written to the output.
    - `numeric-variable` specifies the name of a numeric variable. The value of the numeric variable from the input data set is written to the output.

**Requirement**

You must enclose `string` in quotation marks.

**Example**

```
obj.href( data: 'Link to SAS Support',
         url: 'http://support.sas.com');
```
URL: 'URL-path'
  specifies the fully qualified URL path.

Example  obj.href(data: "Link to SAS Support",
                url: "http://support.sas.com");

Optional Arguments

FORMAT: <character-variable | 'string'>
  specifies the format to be applied to the data argument.

  character-variable  specifies the name of a character variable. The value of the
                     character variable from the input data set is written to the
                     output.

  string-variable    specifies the name of a user-defined format or a format
                     supplied by SAS.

Default    If the data is numeric and no format has been specified, then the data
            value is formatted using the BEST. format.

Requirement You must enclose string in quotation marks.

Example    exempl.href(data: 'Link to SAS Support',
                url: 'http://support.sas.com',
                format: '$upcase10.');

INLINE_ATTR: <character-variable | 'style-attribute'>
  specifies the style attributes to override those defined in the selected style element.
  The style attributes specified by the INLINE_ATTR argument apply only to the
  preceding DATA argument.

  character-variable  specifies the name of a variable from the input data set.

  style-attribute    specifies a style attribute.

Requirement You must enclose style-attribute in quotation marks.

See For a list of style attributes, see “Style Attributes” in SAS Output

Example    exempl.href(data: 'Link to SAS Support',
                url: 'http://support.sas.com',
                inline_attr: 'color=red');

INLINE_ELEM: <character-variable | 'style-element'>
  specifies the style element that contains the collection of style attributes to be applied
  to the data value. The style element specified by the INLINE_ELEM argument
  applies only to the preceding DATA argument.

  character-variable  specifies the name of a variable from the input data set.

  ‘style-element’    specifies a style element.

Default    The default style element for the HREF method is Text.

Requirement You must enclose style-element in quotation marks.
JUST: 'C' | 'L' | 'R' | `char-variable`

specifies the horizontal justification for the data value.

<table>
<thead>
<tr>
<th></th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>specifies center justification.</td>
</tr>
<tr>
<td>L</td>
<td>specifies left justification.</td>
</tr>
<tr>
<td>R</td>
<td>specifies right justification.</td>
</tr>
<tr>
<td>char-variable</td>
<td>specifies the name of a character variable from the input data set.</td>
</tr>
</tbody>
</table>

Default: C

Requirement: You must enclose C, L, and R in quotation marks.

SPLIT: `<character-variable | 'string'>`

specifies the split character to be applied to the data value. A new line will be started when it reaches the specified split character, and will continue on the next line. The split character itself is not part of the data value.

<table>
<thead>
<tr>
<th></th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>character-variable</td>
<td>is the name of a character variable from the input data set.</td>
</tr>
<tr>
<td>string</td>
<td>is one or more characters.</td>
</tr>
</tbody>
</table>

STYLE_ATTR: `<character-variable | 'style-attribute'>`

specifies the style attributes to override those defined in the selected style element. The specified style attributes then apply to all of the text specified in the method, no matter where STYLE_ATTR is specified within the method.

<table>
<thead>
<tr>
<th></th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>character-variable</td>
<td>specifies the name of a variable from the input data set.</td>
</tr>
<tr>
<td>style-attribute</td>
<td>specifies a style attribute.</td>
</tr>
</tbody>
</table>

Requirement: You must enclose style-attribute in quotation marks.

See: For a list of style attributes, see “Style Attributes” in SAS Output Delivery System: User's Guide.

STYLE_ELEM: `<character-variable | 'style-element'>`

specifies the style element that contains the collection of style attributes to be applied to the data value. The style element then applies to all of the text specified in the method, no matter where STYLE_ELEM is specified within the method.

<table>
<thead>
<tr>
<th></th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>character-variable</td>
<td>specifies the name of a variable from the input data set.</td>
</tr>
<tr>
<td>style-element</td>
<td>specifies a style element.</td>
</tr>
</tbody>
</table>

Default: The default style element for the HREF method is Text.

Requirement: You must enclose style-element in quotation marks.

VJUST: `<B|'M'|'T'|character-variable`

specifies the vertical justification for the data value.

B specifies bottom justification
M specifies middle justification
T specifies top justification
char-variable specifies the name of a character variable from the input data set.

Default ‘T’

Requirement You must enclose B, M, and T in quotation marks.

Example

```javascript
obj.href(data: 'Link to SAS Support',
        url: 'http://support.sas.com'), vjust='m';
```

### IMAGE Method

Inserts an image into all open output destinations.

**Valid in:** DATA step  
**Category:** General Purpose  
**Example:** `obj.image(file: 'star.jpg');`

**Syntax**

```
IMAGE (FILE:"fileref" | "filename"<,option-1><,option-2,...>);
```

**Required Argument**

**FILE:** "fileref" | "filename"

inserts an image into all open output destinations.

filename is the name of an external image file.
fileref is a file reference that has been assigned to an external file. Use the FILENAME statement to assign a fileref.

Requirement fileref | filename must be enclosed in quotation marks.

See For information about the FILENAME statement, see SAS Statements: Reference.

**Example**

```javascript
obj.image(file: 'c:\image.jpg');
```

**Optional Arguments**

**STYLE_ATTR:** `<character-variable |'style-attribute'>`

specifies the style attributes to override those defined in the selected style element. The specified style attributes then apply to all of the text specified in the method, no matter where STYLE_ATTR is specified within the method.
**STYLE_ELEM:** `<character-variable> | 'style-element>'`

specifies the style element that contains the collection of style attributes to be applied to the data value. The style element then applies to all of the text specified in the method, no matter where STYLE_ELEM is specified within the method.

- **character-variable** specifies the name of a variable from the input data set.
- **style-element** specifies a style element.

**Default**
The default style element for the IMAGE method is Text.

**Requirement**
You must enclose **style-element** in quotation marks.

**See**
For a list of style elements, see “Style Elements” in SAS Output Delivery System: User's Guide.

---

**LAYOUT_ABSOLUTE Method**

Creates an absolute layout container. The LAYOUT_ABSOLUTE method enables you to specify the exact location on a page for your regions and is limited to one page of layout.

- **Valid in:** DATA step
- **Category:** Layout: Absolute
- **Requirement:** The LAYOUT_ABSOLUTE method must be used with the LAYOUT_END method.
- **ODS destination:** Absolute layout methods are valid in PRINTER destinations only.

**Example:**
```r
trt.layout_absolute();
trt.region(y: '5in');
trt.format_text(data: 'Executive Prospectus', just: 'c',
    style_attr:'font_size=36pt color=cxbbb2e0');
trt.layout_end();
```

**Example:** “Example 12: Creating a Cover Page with the LAYOUT_ABSOLUTE Method” on page 243

---

**Syntax**

`LAYOUT_ABSOLUTE(<option-1>,<option-2, ...>);`

**Optional Arguments**

**HEIGHT:** `'dimension'`

specifies the vertical height of the layout container. The height specified by the HEIGHT argument begins at the point specified by the Y argument and extends down.
**dimension**

is a nonnegative number followed by an optional unit of measure. It is not recommended that you use pixels because of adverse dependencies on resolution that can differ between destinations.

The following units of measure are valid:

- cm centimeters
- em standard typesetting measurements unit for width
- ex standard typesetting measurements unit for height
- in inches
- mm millimeters
- px pixels
- pt a printer’s point

**Default**

If omitted, the HEIGHT argument defaults to the maximum vertical space available on the page.

**Requirement**

*dimension* must be enclosed in quotation marks.

**Example**

```
obj.layout_absolute(height: '10in');
```

**STYLE_ATTR:** `<character-variable | 'style-attribute'>`

specifies the style attributes to override those defined in the selected style element. The specified style attributes then apply to all of the text specified in the method, no matter where STYLE_ATTR is specified within the method.

- **character-variable** specifies the name of a variable from the input data set.
- **style-attribute** specifies a style attribute.

**Requirement**

You must enclose **style-attribute** in quotation marks.

**See**

For a list of style attributes, see “Style Attributes” in *SAS Output Delivery System: User's Guide*.

**STYLE_ELEM:** `<character-variable | 'style-element'>`

specifies the style element that contains the collection of style attributes to be applied to the data value. The style element then applies to all of the text specified in the method, no matter where STYLE_ELEM is specified within the method.

- **character-variable** specifies the name of a variable from the input data set.
- **style-element** specifies a style element.

**Default**

The default style element for the LAYOUT_ABSOLUTE method is LayoutContainer.

**Requirement**

You must enclose **style-element** in quotation marks.

**See**

For a list of style elements, see “Style Elements” in *SAS Output Delivery System: User's Guide*. 
**WIDTH: 'dimension'**

specifies the horizontal width of the layout container. The width specified by the WIDTH argument begins at the point specified by the X argument and extends to the right.

*dimension*

is a nonnegative number followed by an optional unit of measure. It is not recommended that you use pixels because of adverse dependencies on resolution that can differ between destinations.

The following units of measure are valid:

- cm centimeters
- em standard typesetting measurements unit for width
- ex standard typesetting measurements unit for height
- in inches
- mm millimeters
- px pixels
- pt a printer’s point

**Default**

If omitted, the WIDTH argument defaults to the maximum horizontal space available on the page.

**Requirement**

*dimension* must be enclosed in quotation marks.

**Example**

```javascript
obj.layout_absolute(width: '5in');
```

**X: 'dimension'**

specifies the horizontal starting point for the layout container. By default, the starting point for a layout is the top left corner, underneath any specified titles. If there are no titles, then the default is the corner that is set up by the top and left page margin. The position for the top left corner is \( x=0 \mid y=0 \). An X argument specification moves the starting point to the right. That will then be the new horizontal starting point for the layout container. The width specified by the WIDTH argument begins at the point specified by the X argument and extends to the right.

*dimension*

is a nonnegative number followed by an optional unit of measure. It is not recommended that you use pixels because of adverse dependencies on resolution that can differ between destinations.

The following units of measure are valid:

- cm centimeters
- em standard typesetting measurements unit for width
- ex standard typesetting measurements unit for height
- in inches
- mm millimeters
- px pixels
- pt a printer’s point
Default

By default, the starting point for a layout is the top left corner. The position for the top left corner is x=0 | y=0.

Requirement

*dimension* must be enclosed in quotation marks.

Example

```ruby
obs.layout_absolute(y: '3in', x: '3in');
```

**Y: 'dimension'**

specifies the vertical starting point for the layout container. By default, the starting point for a layout is the top left corner, underneath any specified titles. If there are no titles, then the default is the corner that is set up by the top and left page margin. The position for the top left corner is x=0 | y=0. A Y argument specification moves the starting point down. That will then be the new vertical starting point for the layout container. The height specified by the HEIGHT argument begins at the point specified by the Y argument and extends down.

*dimension*

is a nonnegative number followed by an optional unit of measure. It is not recommended that you use pixels because of adverse dependencies on resolution that can differ between destinations.

The following units of measure are valid:

- cm  centimeters
- em  standard typesetting measurements unit for width
- ex  standard typesetting measurements unit for height
- in  inches
- mm  millimeters
- px  pixels
- pt  a printer’s point

Default

By default, the starting point for a layout is the top left corner. The position for the top left corner is x=0 | y=0.

Requirement

*dimension* must be enclosed in quotation marks.

Example

```ruby
obs.layout_absolute(y: '3in', x: '3in');
```

**Details**

Absolute layouts are limited to a single page of output. If the output is too large to fit in the fixed-size container, the output is discarded or partially displayed, and you receive a warning in your log.

If you want to create output on two or more pages, you must close the LAYOUT_ABSOLUTE method block on the first page and create a new LAYOUT_ABSOLUTE method block on each successive page. If you do not want to start a new DATA step, you can use the PAGE method.

```ruby
obj.layout_end();
obj.page();
obj.layout_absolute();
```
**LAYOUT_GRIDDED Method**

Creates a gridded layout container. The LAYOUT_GRIDDED method enables you to arrange output in a two-dimensional gridded structure such as a spreadsheet or a piece of graph paper.

**Valid in:**  
DATA step

**Category:**  
Layout: Gridded

**Requirement:**  
The LAYOUT_GRIDDED method must be used with the LAYOUT_END method.

**ODS destination:**  
Gridded layout methods are valid in the HTML and PRINTER destinations only.

**See:**  
For information about terminology and concepts about layout methods, see “Terminology” on page 112.

**Example:**  
```plaintext
obj.layout_gridded(columns: 3, row_gutter: '1mm', column_gutter: '1mm');
obj.region();
obj.format_text(data: '25', style_attr: 'color=cxbbb2e0 font_size=72pt');
obj.region();
obj.format_text(data: '%*off', split: '*',
    style_attr: "color=cxbbb2e0 font_size=32pt");
obj.region();
obj.format_text(data: 'Now *through *March 25', split: '*',
    style_attr: 'color=cxbbb2e0 font_size=10pt');
obj.layout_end();
```

**Syntax**

```plaintext
LAYOUT_GRIDDED(<option-1>,<option-2, ...>);
```

**Optional Arguments**

**COLUMNS: number | numeric variable**  
specifies the fixed number of columns in a gridded layout.

- `number` is a positive integer.
- `numeric-variable` is the name of a numeric variable from the input data set.

**Default**  
1

**Example**  
```plaintext
obj.layout_gridded(columns: 3, column_widths: '1in',
    column_widths: '2in', column_widths: '3in');
```

**COLUMN_GUTTER: 'dimension'**  
specifies the horizontal space between each column.

- `dimension` is a nonnegative number followed by an optional unit of measure. It is not recommended that you use pixels because of adverse dependencies on resolution that can differ between destinations.

The following units of measure are valid:

- `cm` centimeters
- `em` standard typesetting measurements unit for width
ex  standard typesetting measurements unit for height
in  inches
mm  millimeters
px  pixels
pt  a printer’s point

Requirement  *dimension* must be enclosed in quotation marks.

Example  
```javascript
obj.layout_gridded(
columns: 3, row_gutter: '1mm',
column_gutter: '1mm');
```

**COLUMN_WIDTHS: 'dimension'**
specifies the column width. If you have multiple columns, specify a comma-delimited list of COLUMN_WIDTHS arguments. Each COLUMN_WIDTHS argument corresponds to a column in the order in which it is specified. For example, the following code would result in three columns. Column one would be one inch, column 2 would be two inches, and column three would be three inches.

```javascript
obj.layout_gridded(columns: 3, column_widths:'1in',
                    column_widths:'2in', column_widths:'3in');
```

You can use the COLUMN_WIDTHS argument only when the COLUMNS argument is specified also.

dimension  
is a nonnegative number followed by an optional unit of measure. It is not recommended that you use pixels because of adverse dependencies on resolution that can differ between destinations.

The following units of measure are valid:

- cm    centimeters
- em    standard typesetting measurements unit for width
- ex    standard typesetting measurements unit for height
- in    inches
- mm    millimeters
- px    pixels
- pt    a printer’s point

**Restrictions**  The number of column widths specified must match the number of columns specified, or a warning will be produced, and the argument will be ignored.

You can only use the COLUMN_WIDTHS argument when the COLUMNS argument is specified also.

**Requirement**  *dimension* must be enclosed in quotation marks.

**HEIGHT: 'dimension'**
specifies the vertical height of the layout container. The height specified by the HEIGHT argument begins at the point specified by the Y argument and extends down.
**dimension**

is a nonnegative number followed by an optional unit of measure. It is not recommended that you use pixels because of adverse dependencies on resolution that can differ between destinations.

The following units of measure are valid:

- **cm** centimeters
- **em** standard typesetting measurements unit for width
- **ex** standard typesetting measurements unit for height
- **in** inches
- **mm** millimeters
- **px** pixels
- **pt** a printer’s point

**Default**

If omitted, the **HEIGHT** argument defaults to the maximum vertical space needed to display all regions.

**Requirement**

*dimension* must be enclosed in quotation marks.

**Example**

```javascript
obj.layout_gridded(x: '3.8in', height: '4in');
```

**ORDER_TYPE: COLUMN_MAJOR | ROW_MAJOR**

populates the grid by rows or columns.

**COLUMN_MAJOR**

specifies that the gridded layout first populates all regions in the first column before moving on the next column.

**Example**

```javascript
obj.layout_gridded(order_type: 'column_major');
```

**ROW_MAJOR**

specifies that the gridded layout first populates all regions in the row before moving on the next row.

**Example**

```javascript
obj.layout_gridded(order_type: 'row_major');
```

**Default**

ROW_MAJOR

**Restriction**

**ORDER_TYPE** is valid only in PRINTER destinations.

**ROWS: number | numeric variable**

specifies the fixed number of rows in the gridded layout.

**number** is a positive integer.

**numeric-variable** is the name of a numeric variable from the input data set.

**Default**

If omitted, the **ROWS** argument defaults to the maximum number of regions created in the vertical direction. If there are two columns, then the number of rows is half of the number of regions.

**Example**

```javascript
obj.layout_gridded(rows: 3);
```
ROW_GUTTER: 'dimension'
specifies the vertical space between each row.

dimension
is a nonnegative number followed by an optional unit of measure. It is not recommended that you use pixels because of adverse dependencies on resolution that can differ between destinations.

The following units of measure are valid:

    cm  centimeters
    em  standard typesetting measurements unit for width
    ex  standard typesetting measurements unit for height
    in  inches
    mm  millimeters
    px  pixels
    pt  a printer’s point

Requirement  *dimension* must be enclosed in quotation marks.

Example  
```
obj.layout_gridded(columns: 3, row_gutter: '1mm',
                    column_gutter: '1mm');
```

ROW_HEIGHTS: 'dimension'
specifies the height of each row. If you have multiple rows, specify a comma-delimited list of ROW_HEIGHTS arguments. Each ROW_HEIGHTS argument corresponds to a row in the order in which it is specified. For example, the following code would result in three rows. Row one would be one inch, row 2 would be two inches, and row three would be three inches.

```
obj.layout_gridded(rows: 3, row_heights:'1in',
                    row_heights:'2in', row_heights:'3in');
```

You can use the ROW_HEIGHTS argument only when the ROWS argument is specified also.

dimension
is a nonnegative number followed by an optional unit of measure. It is not recommended that you use pixels because of adverse dependencies on resolution that can differ between destinations.

The following units of measure are valid:

    cm  centimeters
    em  standard typesetting measurements unit for width
    ex  standard typesetting measurements unit for height
    in  inches
    mm  millimeters
    px  pixels
    pt  a printer’s point

Restrictions  The number of row heights must match the number of rows specified or else a warning will be produced, and the option will be ignored.
You can specify the ROW_HEIGTHS argument only if the ROWS argument is also specified.

**STYLE_ATTR:** `<character-variable | 'style-attribute'>`
specifies the style attributes to override those defined in the selected style element. The specified style attributes then apply to all of the text specified in the method, no matter where STYLE_ATTR is specified within the method.

- `character-variable` specifies the name of a variable from the input data set.
- `style-attribute` specifies a style attribute.

**Requirement**
You must enclose `style-attribute` in quotation marks.

**See**
For a list of style attributes, see “Style Attributes” in *SAS Output Delivery System: User's Guide*.

**STYLE_ELEM:** `<character-variable | 'style-element'>`
specifies the style element that contains the collection of style attributes to be applied to the data value. The style element then applies to all of the text specified in the method, no matter where STYLE_ELEM is specified within the method.

- `character-variable` specifies the name of a variable from the input data set.
- `style-element` specifies a style element.

**Default**
The default style element for the LAYOUT_ABSOLUTE method is LayoutContainer.

**Requirement**
You must enclose `style-element` in quotation marks.

**See**
For a list of style elements, see “Style Elements” in *SAS Output Delivery System: User's Guide*.

**WIDTH:** `'dimension'`
specifies the horizontal width of the layout container. The width specified by the WIDTH argument begins at the point specified by the X argument and extends to the right.

- `dimension` is a nonnegative number followed by an optional unit of measure. It is not recommended that you use pixels because of adverse dependencies on resolution that can differ between destinations.

The following units of measure are valid:

- `em` centimeters
- `em` standard typesetting measurements unit for width
- `ex` standard typesetting measurements unit for height
- `in` inches
- `mm` millimeters
- `px` pixels
- `pt` a printer’s point
Default

If omitted, the WIDTH argument defaults to the maximum horizontal space needed to display all regions.

Requirement

*dimension* must be enclosed in quotation marks.

Example

```javascript
obj.layout_gridded(x: '3.8in', width: '2in', height: '4in');
```

**X: 'dimension'**

specifies the horizontal starting point for the layout container. When using the X argument, the starting point for a layout is the top left corner, underneath the title. The position for the top left corner is x=0 | y=0. An X argument specification moves the starting point to the left. That will be the new horizontal starting point for the layout container. The width specified by the WIDTH argument begins at the point specified by the X argument and extends to the right.

*dimension*

is a nonnegative number followed by an optional unit of measure. It is not recommended that you use pixels because of adverse dependencies on resolution that can differ between destinations.

The following units of measure are valid:

- cm centimeters
- em standard typesetting measurements unit for width
- ex standard typesetting measurements unit for height
- in inches
- mm millimeters
- px pixels
- pt a printer’s point

Default

By default, a gridded layout container is centered on the page.

Restriction

The X argument is valid only in PRINTER destinations.

Requirement

*dimension* must be enclosed in quotation marks.

Example

```javascript
obj.layout_gridded(x: '3.8in', width: '2in', height: '4in');
```

**Y: 'dimension'**

specifies the vertical starting point for the layout container. When using the Y argument, the starting point for a layout is the top left corner, underneath the title. The position for the top left corner is x=0 | y=0. A Y argument specification moves the starting point down. That will be the new vertical starting point for the layout container. The height specified by the HEIGHT argument begins at the point specified by the Y argument and extends down.

*dimension*

is a nonnegative number followed by an optional unit of measure. It is not recommended that you use pixels because of adverse dependencies on resolution that can differ between destinations.
The following units of measure are valid:

- **cm** centimeters
- **em** standard typesetting measurements unit for width
- **ex** standard typesetting measurements unit for height
- **in** inches
- **mm** millimeters
- **px** pixels
- **pt** a printer’s point

**Default** By default, a gridded layout container is centered on the page.

**Restriction** The Y argument is valid only in PRINTER destinations.

**Requirement** *dimension* must be enclosed in quotation marks.

**Comparisons**
Gridded layouts can span any number of pages and are valid for PRINTER and HTML destinations only.

---

**LAYOUT_END Method**
Ends a gridded or absolute layout block.

**Valid in:** DATA step

**Categories:**
- Layout: Absolute
- Layout: Gridded

**Requirement:** The LAYOUT_END method must be used with the LAYOUT_GRIDDED or LAYOUT_ABSOLUTE method.

**See:** For information about terminology and concepts about layout methods, see “Terminology” on page 112.

**Syntax**

```
LAYOUT_END( );
```

---

**LINE Method**
Inserts a horizontal rule (line) across the page.

**Valid in:** DATA step

**Category:** General Purpose

**Restriction:** A line cannot be drawn within a table. The LINE method cannot be specified between the TABLE_START and TABLE_END methods.

**Example:**

```
obj.line();
```
Syntax

LINE(<option-1>,<option-2>, ...);

Optional Arguments

STYLE_ATTR: <character-variable | 'style-attribute'>
specifies the style attributes to override those defined in the selected style element. The specified style attributes then apply to all of the text specified in the method, no matter where STYLE_ATTR is specified within the method.

character-variable specifies the name of a variable from the input data set.

style-attribute specifies a style attribute.

Requirement You must enclose style-attribute in quotation marks.

See For a list of style attributes, see “Style Attributes” in SAS Output Delivery System: User's Guide.

STYLE_ELEM: <character-variable | 'style-element'>
specifies the style element that contains the collection of style attributes to be applied to the data value. The style element then applies to all of the text specified in the method, no matter where STYLE_ELEM is specified within the method.

character-variable specifies the name of a variable from the input data set.

style-element specifies a style element.

Default The default style element for the LINE method is Text.

Requirement You must enclose style-element in quotation marks.

See For a list of style elements, see “Style Elements” in SAS Output Delivery System: User's Guide.

SIZE: 'dimension'
specifies the thickness of the line.

dimension is a nonnegative number followed by an optional unit of measure. It is not recommended that you use pixels because of adverse dependencies on resolution that can differ between destinations.

The following units of measure are valid:

cm centimeters

cm standard typesetting measurements unit for width

ex standard typesetting measurements unit for height

in inches

mm millimeters

px pixels

pt a printer’s point

Restriction The SIZE argument is only valid for PRINTER destinations.
Requirement  *dimension* must be enclosed in quotation marks.

Example  
```javascript
obj.line(size: '1mm');
```

---

### NOTE Method

Writes a note to the active output destination(s).

**Valid in:** DATA step  
**Category:** Text  
**Example:**  
```javascript
obj.note(data: strip(street),
    style_attr: 'font_style=italic
    font_size=12pt
    font_weight=bold
    color=cxbbb2e0');
```

**Examples:**  
“Example 1: Formatting Data with Style Attributes and Style Elements” on page 195  
“Example 4: Creating a Flyer with Text Methods” on page 206  
“Example 5: Working with Data Sets” on page 210  
“Example 6: Working with BY Groups and IF-THEN Processing” on page 213

---

### Syntax

```javascript
NOTE(DATA: displayed-data <option-1><option-2,...>);
```

### Required Argument

**DATA:** <"string" | number | character-variable | numeric-variable>  
specifies the data to display.

*string* specifies a text string.

number specifies an integer.

character-variable specifies the name of a character variable. The value of the character variable from the input data set is written to the output.

numeric-variable specifies the name of a numeric variable. The value of the numeric variable from the input data set is written to the output.

**Requirement** You must enclose *string* in quotation marks.

**Example**  
```javascript
obj.note(data: 'Here is a note.');
```

---

### Optional Arguments

**FORMAT:** <character-variable | 'string'>  
specifies the format to be applied to the data argument.
character-variable specifies the name of a character variable. The value of the character variable from the input data set is written to the output.

string-variable specifies the name of a user-defined format or a format supplied by SAS.

Default If the data is numeric and no format has been specified, then the data value is formatted using the BEST format.

Requirement You must enclose string in quotation marks.

Example `exmpl.note(data: exdate, format: 'date9.');`

**INLINE_ATTR:** `<character-variable> | 'style-attribute'>`
specifies the style attributes to override those defined in the selected style element. The style attributes specified by the INLINE_ATTR argument apply only to the preceding DATA argument.

Parameter:
- character-variable specifies the name of a variable from the input data set.
- style-attribute specifies a style attribute.

Requirement You must enclose style-attribute in quotation marks.

See For a list of style attributes, see “Style Attributes” in SAS Output Delivery System: User’s Guide.

Example `exmpl.note(data: 'Make this text red', inline_attr: 'color=red', data: 'and this bold', inline_attr: 'font_weight=bold', data: 'and use a 16pt font', inline_attr: 'font_size=16pt');`

**INLINE_ELEM:** `<character-variable> | 'style-element'>`
specifies the style element that contains the collection of style attributes to be applied to the data value. The style element specified by the INLINE_ELEM argument applies only to the preceding DATA argument.

Parameter:
- character-variable specifies the name of a variable from the input data set.
- style-element specifies a style element.

Default The default style element for the NOTE method is Note.

Requirement You must enclose style-element in quotation marks.

See For a list of style elements, see “Style Elements” in SAS Output Delivery System: User’s Guide.

Example `exmpl.note(data: 'This text is formatted using the INLINE_ELEM argument and the Output Style Element.', inline_elem: 'Output', data: 'The INLINE_ELEM argument applies only to the preceding text.');`
JUST: 'C' | 'L' | 'R' | char-variable
specifies the horizontal justification for the data value.

C specifies center justification.
L specifies left justification.
R specifies right justification.
char-variable specifies the name of a character variable from the input data set.

Default C

Requirement You must enclose C, L, and R in quotation marks.

SPLIT: <character-variable | 'string'>
specifies the split character to be applied to the data value. A new line will be started when it reaches the specified split character, and will continue on the next line. The split character itself is not part of the data value.

caller-variable is the name of a character variable from the input data set. The character variable’s value resolves to the character.
string is one or more characters.

Example obj.note(data: 'Line 1#Line 2', split: '#');

STYLE_ATTR: <character-variable | 'style-attribute'>
specifies the style attributes to override those defined in the selected style element. The specified style attributes then apply to all of the text specified in the method, no matter where STYLE_ATTR is specified within the method.

caller-variable specifies the name of a variable from the input data set.
style-attribute specifies a style attribute.

Requirement You must enclose style-attribute in quotation marks.

See For a list of style attributes, see “Style Attributes” in SAS Output Delivery System: User's Guide.

Example exmpl.note(data: 'With the STYLE_ATTR argument, ', style_attr: 'color=red', data: 'the style attribute applies to all text.');

STYLE_ELEM: <character-variable | 'style-element'>
specifies the style element that contains the collection of style attributes to be applied to the data value. The style element then applies to all of the text specified in the method, no matter where STYLE_ELEM is specified within the method.

caller-variable specifies the name of a variable from the input data set.
style-element specifies a style element.

Default The default style element for the NOTE method is Note.

Requirement You must enclose style-element in quotation marks.
See For a list of style elements, see “Style Elements” in SAS Output Delivery System: User's Guide.

**VJUST:** `<B'| 'M' | 'T' | character-variable>`
specifies the vertical justification for the data value.

- B specifies bottom justification
- M specifies middle justification
- T specifies top justification

*char-variable* specifies the name of a character variable from the input data set.

**Default** 'T'

**Requirement** You must enclose B, M, and T in quotation marks.

**Example**
```sas
exmpl.note(data: 'Here is a note.',
           style_attr: '3in', vjust: 'B');
```

**URL:** `URL-path`
specifies the fully qualified URL path.

**Example**
```sas
obj.note(data: "Link to SAS Support",
         url: "http://support.sas.com");
```

---

### PAGE Method

Creates a page break.

**Valid in:** DATA step  
**Category:** Page

**Syntax**
```
PAGE();
```

### REGION Method, Absolute

Creates a region container for absolute layouts.

**Valid in:** DATA step  
**Category:** Layout: Absolute  
**ODS destination:** Absolute layout methods are valid in PRINTER destinations only.

**Tip:** Region containers can overlap. If you specify regions without any width or height arguments, the two region will overlap.

**Example**
```sas
trt.layout_absolute();
trt.region(y: '5in');
trt.format_text(data: 'Executive Prospectus', just: 'c',
```
Example: “Example 12: Creating a Cover Page with the LAYOUT_ABSOLUTE Method” on page 243

Syntax
REGION(<option-1><,option-2, ...>);

Optional Arguments
HEIGHT: 'dimension'
specifies the vertical width of the region.

dimension
is a nonnegative number followed by an optional unit of measure. It is not
recommended that you use pixels because of adverse dependencies on resolution
that can differ between destinations.

The following units of measure are valid:

- cm  centimeters
- em  standard typesetting measurements unit for width
- ex  standard typesetting measurements unit for height
- in  inches
- mm  millimeters
- px  pixels
- pt  a printer’s point

Default
The default height is all of the available space in the container.

Restrictions
The height is restricted by the dimensions of the layout.
The sum of all region heights cannot exceed the vertical dimension of
the layout.

Requirement
dimension must be enclosed in quotation marks.

Example
obj.region(height: '4in');

STYLE_ATTR: <character-variable | 'style-attribute'>
specifies the style attributes to override those defined in the selected style element.
The specified style attributes then apply to all of the text specified in the method, no
matter where STYLE_ATTR is specified within the method.

character-variable  specifies the name of a variable from the input data set.
style-attribute      specifies a style attribute.

Requirement
You must enclose style-attribute in quotation marks.

See
For a list of style attributes, see “Style Attributes” in SAS Output
STYLE_ELEM: `<character-variable | 'style-element'>`

specifies the style element that contains the collection of style attributes to be applied to the data value. The style element then applies to all of the text specified in the method, no matter where STYLE_ELEM is specified within the method.

**character-variable** specifies the name of a variable from the input data set.

**style-element** specifies a style element.

**Requirement** You must enclose **style-element** in quotation marks.

**See** For a list of style elements, see “Style Elements” in *SAS Output Delivery System: User's Guide*.

WIDTH: `'dimension'`

specifies the horizontal width of the region.

**dimension**

is a nonnegative number followed by an optional unit of measure. It is not recommended that you use pixels because of adverse dependencies on resolution that can differ between destinations.

The following units of measure are valid:

- cm centimeters
- em standard typesetting measurements unit for width
- ex standard typesetting measurements unit for height
- in inches
- mm millimeters
- px pixels
- pt a printer’s point

**Default** The default width is all of the available space in the container.

**Restriction** The width is restricted by the dimensions of the layout.

**Requirement** **dimension** must be enclosed in quotation marks.

**Example**

```javascript
obj.region(width: '5in');
```

X: `'dimension'`

specifies the horizontal start position of the region on a page.

**dimension**

is a nonnegative number followed by an optional unit of measure. It is not recommended that you use pixels because of adverse dependencies on resolution that can differ between destinations.

The following units of measure are valid:

- cm centimeters
- em standard typesetting measurements unit for width
- ex standard typesetting measurements unit for height
- in inches
Y: 'dimension'
specifies the vertical start position of the region on a page.

dimension
is a nonnegative number followed by an optional unit of measure. It is not recommended that you use pixels because of adverse dependencies on resolution that can differ between destinations.

The following units of measure are valid:

- cm centimeters
- em standard typesetting measurements unit for width
- ex standard typesetting measurements unit for height
- in inches
- mm millimeters
- px pixels
- pt a printer’s point

Details
A region container is an area that contains output such as text, tables, and images. Region containers can be nested and can overlap. If you use regions without any width or height arguments specified, the two region will overlap. The REGION method must be used within a LAYOUT_ABSOLUTE method block or a LAYOUT_GRIDDED method block.
The following graphic illustrates the layout sections that the RWI can create and modify:

**Output 11.1  Layout Sections**

![Diagram of layout sections]

**REGION Method, Gridded**

Creates a region container for gridded layouts.

- **Valid in:** DATA step
- **Category:** Layout: Gridded
- **ODS destination:** Gridded layout methods are valid in the HTML and PRINTER destinations only.
- **See:** For information about terminology and concepts about layout methods, see “Terminology” on page 112.

**Example:**

```javascript
obj.layout_gridded(columns: 2);
obj.region(width: '3.25in');

obj.format_text(data: strip(street),
    style_attr: 'font_size=14pt width=100pct just=left');

obj.format_text(data: strip(city) || ', ' || state || ' ' || zip,
    style_attr: 'font_size=14pt width=100pct just=left');

obj.format_text(data: put(today(), worddate18.),
    style_attr: 'font_size=14pt width=100pct just=left');

obj.region();

obj.image(file: 'star.jpg');

obj.layout_end();
```
Syntax

REGION(<option-1>, <option-2>, ...);

Optional Arguments

COLUMN: number | numeric-variable
specifies the current grid column position in the gridded layout. The GRIDDED_LAYOUT method automatically tracks the current grid column position, and the position is incremented for every REGION method.

- number is a positive integer.
- numeric-variable is the name of a numeric variable from the input data set.

Restrictions

After you have skipped a grid column, you cannot go back to it.

Tip

The COLUMN argument is useful when you want to skip regions in the gridded layout.

Example

exmpl.region(column: 5, column_span: 3);

COLUMN_SPAN: number | numeric-variable
specifies the number of grid columns that the region will occupy. The COLUMN_SPAN argument enables you to combine adjacent grid columns in a gridded layout.

- number is a positive integer.
- numeric-variable is the name of a numeric variable from the input data set.

Default

1

Example

exmpl.region(column: 5, column_span: 3);

HEIGHT: 'dimension'
specifies the vertical width of the region.

dimension
is a nonnegative number followed by an optional unit of measure. It is not recommended that you use pixels because of adverse dependencies on resolution that can differ between destinations.

The following units of measure are valid:

- cm centimeters
- em standard typesetting measurements unit for width
- ex standard typesetting measurements unit for height
- in inches
- mm millimeters
- px pixels
- pt a printer’s point
Default
If you omit the HEIGHT argument, the default is the maximum vertical space needed to display the output contained in the region.

Restrictions
The height is restricted by the dimensions of the layout.
The sum of all region heights cannot exceed the vertical dimension of the layout.

Requirement
*dimension* must be enclosed in quotation marks.

Example
```
obj.region(height: '4in');
```

**ROW: number | numeric variable**
specifies the current grid row position in the gridded layout. The GRIDDED_LAYOUT method automatically tracks the current row position, and the position is incremented for every REGION method.

*number* is a positive integer.

*numeric-variable* is the name of a numeric variable from the input data set.

Restriction
After you have skipped a grid row, you cannot go back to it.

Tip
The ROW argument is useful when you want to skip regions in the gridded layout.

Example
```
exmpl.region(row: 5, row_span: 3);
```

**ROW_SPAN: number | numeric variable**
specifies the number of grid rows that the region will occupy. The ROW_SPAN argument enables you to combine adjacent grid rows in a gridded layout.

*number* is a positive integer.

*numeric-variable* is the name of a numeric variable from the input data set.

Default
1

Example
```
exmpl.region(row: 2, row_span: 3);
```

**STYLE_ATTR: <character-variable | 'style-attribute'>**
specifies the style attributes to override those defined in the selected style element. The specified style attributes then apply to all of the text specified in the method, no matter where STYLE_ATTR is specified within the method.

*character-variable* specifies the name of a variable from the input data set.

*style-attribute* specifies a style attribute.

Requirement
You must enclose *style-attribute* in quotation marks.

See
For a list of style attributes, see “Style Attributes” in *SAS Output Delivery System: User's Guide*.

**STYLE_ELEM: <character-variable | 'style-element'>**
specifies the style element that contains the collection of style attributes to be applied to the data value. The style element then applies to all of the text specified in the method, no matter where STYLE_ELEM is specified within the method.
character-variable specifies the name of a variable from the input data set.

style-element specifies a style element.

Default The default style element for the REGION method is Text.

Requirement You must enclose style-element in quotation marks.

See For a list of style elements, see “Style Elements” in SAS Output Delivery System: User's Guide.

**WIDTH: 'dimension'**

specifies the horizontal width of the region.

*dimension*

is a nonnegative number followed by an optional unit of measure. It is not recommended that you use pixels because of adverse dependencies on resolution that can differ between destinations.

The following units of measure are valid:

- cm centimeters
- em standard typesetting measurements unit for width
- ex standard typesetting measurements unit for height
- in inches
- mm millimeters
- px pixels
- pt a printer’s point

Default If you omit the WIDTH argument, the default is the maximum horizontal space needed to display the output contained in the region.

Restrictions The width is restricted by the dimensions of the layout.

The sum of all region widths cannot exceed the width of the layout container.

Requirement dimension must be enclosed in quotation marks.

Example obj.region(width: '3.25in');

**Details**

A region container is an area that contains output such as text, tables, and images. Region containers can be nested. The REGION method must be used within a LAYOUT_ABSOLUTE method block or a LAYOUT_GRIDDED method block.
The following graphic illustrates the layout sections that the RWI can create and modify:

**Output 11.2  Layout Sections**

---

**ROW_END Method**

Specifies the end of a row.

- **Valid in:** DATA step
- **Category:** Table
- **Requirement:** The ROW_END method is always used with the ROW_START method.
- **See:** For information about terminology and concepts used to create tables, see “Terminology” on page 112.

**Example:**

```javascript
obj.table_start();
obj.head_start();
obj.row_start();
  obj.format_cell(data: 'A single cell table');
obj.row_end();
obj.head_end();
obj.table_end();
```

**Examples:**

“Example 7: Creating a Simple Table” on page 216
“Example 8: Formatting Cells Using the Textdecoration Style Attribute ” on page 219
“Example 9: Creating a Table with Row and Column Spanning” on page 223
“Example 10: Using the Page Method” on page 228
Syntax

ROW_END();

**ROW_START Method**

Specifies the beginning of a row.

- **Valid in:** DATA step
- **Category:** Table
- **Requirement:** The ROW_START method must be used with the ROW_END method.
- **See:** For information about terminology and concepts used to create tables, see “Terminology” on page 112.

**Example:**

```javascript
obj.table_start();
obj.head_start();
  obj.row_start();
  obj.format_cell(data: 'A single cell table');
  obj.row_end();
obj.head_end();
obj.table_end();
```

**Examples:**

- “Example 7: Creating a Simple Table” on page 216
- “Example 8: Formatting Cells Using the Textdecoration Style Attribute” on page 219
- “Example 9: Creating a Table with Row and Column Spanning” on page 223
- “Example 10: Using the Page Method” on page 228

---

**Syntax**

ROW_START(<option-1>,<option-2, ...>);

**Optional Arguments**

**ROW:** 'index-number'

specifies the row index. The table keeps track of its current row index, allowing you to skip blank rows. After a row is skipped, you cannot go back to it.

- **index-number** is a positive integer.
  - **Restriction** index-number must be enclosed in quotation marks.

**STYLE_ATTR:** <character-variable> | 'style-attribute'>

specifies the style attributes to override those defined in the selected style element. The specified style attributes then apply to all of the text specified in the method, no matter where STYLE_ATTR is specified within the method.

- **character-variable** specifies the name of a variable from the input data set.
- **style-attribute** specifies a style attribute.
  - **Requirement** You must enclose style-attribute in quotation marks.
STYLE_ELEM: `<character-variable | 'style-element'>`

This specifies the style element that contains the collection of style attributes to be applied to the data value. The style element then applies to all of the text specified in the method, no matter where STYLE_ELEM is specified within the method.

- **character-variable**: specifies the name of a variable from the input data set.
- **style-element**: specifies a style element.

**Requirement**

You must enclose `style-element` in quotation marks.

**See**

For a list of style elements, see “Style Elements” in *SAS Output Delivery System: User's Guide*.

**TYPE: 'B' | 'F' | 'H'**

Identifies the table sections.

- **B**: specifies the body (data) section
- **F**: specifies the footer section
- **H**: specifies the header section

**Default**

B

**Tip**

Depending on the type that you choose, specifying the TYPE argument is an alternative to specifying the BODY_START, HEAD_START, or FOOT_START statements (with their respective END statements).

---

**TABLE_END Method**

Specifies the end of a table.

- **Valid in**: DATA step
- **Category**: Table
- **Requirement**: The TABLE_END method is always used with the TABLE_START method.
- **See**: For information about terminology and concepts used to create tables, see “Terminology” on page 112.
- **Example**:

```javascript
obj.table_start();
obj.head_start();
obj.row_start();
  obj.format_cell(data: 'A single cell table');
obj.row_end();
obj.head_end();
obj.table_end();
```
**Syntax**

TABLE_END();

---

**TABLE_START Method**

Specifies the start of a table.

- **Valid in:** DATA step
- **Category:** Table
- **Requirement:** The TABLE_START method must be used with the TABLE_END method.
- **Tip:** After you start a table, you should continue using the table methods exclusively until you close the table.
- **See:** For information about terminology and concepts used to create tables, see "Terminology" on page 112.

**Example:**

```plaintext
obj.table_start();
obj.head_start();
obj.row_start();
obj.format_cell(data: 'A single cell table');
obj.row_end();
obj.head_end();
obj.table_end();
```

**Examples:**

- “Example 7: Creating a Simple Table” on page 216
- “Example 8: Formatting Cells Using the Textdecoration Style Attribute” on page 219
- “Example 9: Creating a Table with Row and Column Spanning” on page 223
- “Example 10: Using the Page Method” on page 228

---

**Syntax**

TABLE_START(<option-1>,<option-2,...>);

**Optional Arguments**

**JUST:** 'C' | 'L' | 'R' | `char-variable`

- Specifies the horizontal justification for the entire table.

  - C specifies center justification.
  - L specifies left justification.
  - R specifies right justification.
  - `char-variable` specifies the name of a character variable from the input data set.
TITLE Method

Adds a title to a page.

Valid in: DATA step
Category: Page
Examples: obj.title(data: 'Here is a new title');

The following example left-justifies the title:

obj.title (data: 'Left Justified Title',
    style_attr: 'just=left');
Syntax

TITLE(DATA:'displayed-data' <,option-1><,option-2, ...>);

Required Argument

DATA: <"string" | number | character-variable | numeric-variable>
specifies the data to display.

'string' specifies a text string.

number specifies an integer.

character-variable specifies the name of a character variable. The value of the character variable from the input data set is written to the output.

numeric-variable specifies the name of a numeric variable. The value of the numeric variable from the input data set is written to the output.

Requirement You must enclose string in quotation marks.

Example

. obj.title(data: 'Here is a Title');

Optional Arguments

CLEAR: line-number clears the title on the specified line.

line-number is the line number.

Range 1–10

Example The following example clears the 'The SAS System' default title:

obj.title(start: 1, clear: 1);

FORMAT: <character-variable | 'string'> specifies the format to be applied to the data argument.

character-variable specifies the name of a character variable. The value of the character variable from the input data set is written to the output.

string-variable specifies the name of a user-defined format or a format supplied by SAS.

Default If the data is numeric and no format has been specified, then the data value is formatted using the BEST. format.

Requirement You must enclose string in quotation marks.

Example example.title(data: clubpoints, format: 'comma.');
INLINE_ELEM: `<character-variable | 'style-element'>`
specifies the style element that contains the collection of style attributes to be applied to the data value. The style element specified by the INLINE_ELEM argument applies only to the preceding DATA argument.

- **character-variable** specifies the name of a variable from the input data set.
- **'style-element'** specifies a style element.

**Default**
The default style element for the TITLE method is SystemTitle.

**Requirement**
You must enclose **style-element** in quotation marks.

**See**
For a list of style elements, see “Style Elements” in SAS Output Delivery System: User’s Guide.

**Example**
exmpl.title(data: 'This text is formatted using the
INLINE_ELEM argument and the Output Style Element.',
INLINE_ELEM: 'Output',
data: 'The INLINE_ELEM argument applies
only to the preceding text.');

INLINE_ATTR: `<character-variable | 'style-attribute'>`
specifies the style attributes to override those defined in the selected style element. The style attributes specified by the INLINE_ATTR argument apply only to the preceding DATA argument.

- **character-variable** specifies the name of a variable from the input data set.
- **style-attribute** specifies a style attribute.

**Requirement**
You must enclose **style-attribute** in quotation marks.

**See**
For a list of style attributes, see “Style Attributes” in SAS Output Delivery System: User’s Guide.

**Example**
exmpl.title(data: 'Make this text red',
INLINE_ATTR: 'color=red',
data: 'and this bold',
INLINE_ATTR: 'font_weight=bold',
data: ' and use a 16pt font',
INLINE_ATTR: 'font_size=16pt');

START: `line-number`
specifies the line the title will be displayed on.

- **line-number** is the line number.

**Range**
1–10

**Example**
obj.title(data: 'This is the Third Title', start: 3);

SPLIT: `<character-variable | 'string'>`
specifies the split character to be applied to the data value. A new line will be started when it reaches the specified split character, and will continue on the next line. The split character itself is not part of the data value.
character-variable is the name of a character variable from the input data set. The character variable’s value resolves to the character.

string is one or more characters.

**STYLE_ATTR:** `<character-variable \| 'style-attribute'>`

specifies the style attributes to override those defined in the selected style element. The specified style attributes then apply to all of the text specified in the method, no matter where STYLE_ATTR is specified within the method.

- **character-variable** specifies the name of a variable from the input data set.
- **style-attribute** specifies a style attribute.

**Requirement**
You must enclose `style-attribute` in quotation marks.

**See**
For a list of style attributes, see “Style Attributes” in SAS Output Delivery System: User's Guide.

**Example**
exmpl.title(data: 'With the STYLE_ATTR argument, ',
    style_attr: 'color=red',
    data: 'the style attribute applies to all text.');

**STYLE_ELEM:** `<character-variable \| 'style-element'>`

specifies the style element that contains the collection of style attributes to be applied to the data value. The style element then applies to all of the text specified in the method, no matter where STYLE_ELEM is specified within the method.

- **character-variable** specifies the name of a variable from the input data set.
- **style-element** specifies a style element.

**Default**
The default style element for the TITLE method is SystemTitle.

**Requirement**
You must enclose `style-element` in quotation marks.

**See**
For a list of style elements, see “Style Elements” in SAS Output Delivery System: User's Guide.

**Example**
exmpl.title(data: 'This text is formatted using STYLE_ELEM and the SystemTitle Style Element.',
    style_elem: 'UserText',
    data: 'The STYLE_ELEM argument applies to the entire method.');

---

**VIDEO Method**

Embeds video into HTML5 or EPUB output.

- **Valid in:** DATA step
- **Category:** Media
- **Restriction:** The VIDEO method can only be used with the ODS HTML5 statement.
- **See:** For complete documentation on the ODS HTML5 statement, see “ODS HTML5 Statement” in SAS Output Delivery System: User's Guide. For complete documentation on the ODS EPUB statement, see “ODS EPUB Statement” in SAS.
Example: ods html5 file="video.html";

data _null_

dcl odsout obj();
obj.video(
    file: "filepath-here.mp4",
    type: "mp4",
    width: "640",
    height: "480",
    preload: "auto",
    autoplay: "yes",
    poster: "wildlife-thumb.jpg",
    loop: "on"
); 
run;

ods html5 close;

Example: For a complete example using the VIDEO method, see “Example 3: Adding a Video to HTML5 Output” on page 203.

Syntax

VIDEO (FILE: "path" | "FILEREF:fileref" <, TYPE:file-type, >,<video-control-option-1>,<video-control-option-2, ...>);

Required Argument

FILE: "path" | "FILEREF:fileref"
specifies the path of the video to insert into HTML5 or EPUB output.

path is the path to an external video file. path can be a URL or a filename with a fully qualified path.

Requirement path must be enclosed in quotation marks.

Example obj.video (file:"http://channel.sas.com/vod/videos/2012/11598_Overview_Final_SASCOM640x360.mp4",
    type: "mp4");

FILEREF:fileref is a file reference that has been assigned to an external file.

To use a FILEREF with the VIDEO method, you must use the URL option in the FILENAME statement, and then specify the fileref with the FILEREF: identifier.

Requirement FILEREF:fileref must be enclosed in quotation marks.


Example filename VidFile url "http://channel.sas.com/vod/videos/2012/"
Tips
A best practice is to always specify the TYPE: option with the FILE: option. If the TYPE: option is not specified, then the browser will determine the file type. All other options will then default to browser-determined values.

To display multiple video players, you must use separate VIDEO methods.

To specify multiple format of the same video file, you can use multiple FILE: arguments. The browser will then play the first file in the list that is a type that it supports. This enables you to support multiple browsers with one VIDEO method.

Optional Arguments

TYPE: "video-type"
specifies the video type. If the TYPE: option is not specified, then the browser will determine the file type. All other options will then default to browser-determined values.

Most browsers support one or more of the following video types:

- MP4
- OGG
- WEBM

Table 11.2  Supported Video Types for Browsers

<table>
<thead>
<tr>
<th>Browser</th>
<th>Video Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Internet Explorer</td>
<td>MP4</td>
</tr>
<tr>
<td>Chrome</td>
<td>MP4, WebM, Ogg</td>
</tr>
<tr>
<td>Firefox</td>
<td>MP4, WebM, Ogg</td>
</tr>
<tr>
<td>Safari</td>
<td>MP4</td>
</tr>
<tr>
<td>Opera 25</td>
<td>MP4, WebM, Ogg</td>
</tr>
</tbody>
</table>

Requirement
If TYPE is specified, it must be specified before the next FILE is specified.

WIDTH: "dimension"
specifies the width of the video.

dimension
is a nonnegative number followed by an optional unit of measure. dimension can be specified in pixels (px) or inches (in). If no unit is specified, pixels are assumed.
HEIGHT: "dimension"
specifies the height of the video.

dimension
is a nonnegative number followed by an optional unit of measure. *dimension* can be specified in pixels (px) or inches (in). If no unit is specified, pixels are assumed.

PRELOAD: "AUTO | NONE | METADATA"
specifies how the video loads in a browser.

AUTO load the video data as soon as the page loads.

NONE load the video data when user presses the play button.

METADATA load information about the video file when the page loads, but not the video itself.

Default AUTO

AUTOPLAY: "OFF | ON"
specifies when the video will begin playing.

OFF start playing the video when the play button is pushed.

Alias NO

ON start playing the video immediately when the page loads.

Alias YES

Default OFF

POSTER: "path"
specifies the location of a still image to use if the video cannot be found.

LOOP:"OFF | ON"
specifies whether the video is continuously replayed.

OFF play the video once.

Alias NO

ON start replaying the video immediately after it ends.

Alias YES

Default OFF

MUTED:"OFF | ON"
controls whether the audio is muted.

OFF the audio is on when the video starts playing.

Alias NO
ON
  the audio is off when the video starts playing.

Alias  YES

Default  OFF

Restriction  The MUTED: option is not supported in Internet Explorer 9 or earlier.

Examples

Example 1: Formatting Data with Style Attributes and Style Elements

Features:  DECLARE statement

Methods
  FORMAT_TEXT
  NOTE

Arguments
  DATA
  STYLE_ELEM
  STYLE_ATTR
  INLINE_ELEM
  INLINE_ATTR

Details
This example demonstrates the following:

• using the FORMAT_TEXT method to add text to your output
• using the NOTE method to add a note to your output
• using style elements to format your text
• using style attributes to format your text

Adding Unformatted Text to Your Output

```plaintext
filename rwiOut ".";
ods html close;
ods html path=rwiOut file="UnformatText.html"
  title "Using the DATA Argument";
  title "Using the DATA Argument";
data _null_
  dcl odsout obj();
    obj.format_text(data: "This text is displayed with the FORMAT_TEXT method.");
  obj.note(data: "This text is displayed with the NOTE method.");
run;
```
Program Description

Create a file reference for the output, set the ODS options, and specify a title. The current working directory is specified in this example.

```sas
filename rwiOut ".";
ods html close;
ods html path=rwiOut file="UnformatText.html"
  title "Using the DATA Argument";
```

Begin the DATA Null step and declare the local variable `obj`. The DATA _NULL_ statement begins the DATA step. The DECLARE statement assigns `obj` as the variable that identifies the object. The DECLARE statement also creates an instance.

```sas
  title "Using the DATA Argument";
data _null_
  dcl odsout obj();
```

Add unformatted text. The FORMAT_TEXT method with the DATA argument specified adds text to all open destinations. The NOTE method with the DATA argument specified adds a note to all open destinations.

```sas
  obj.format_text(data: "This text is displayed with the FORMAT_TEXT method.");
  obj.note(data: "This text is displayed with the NOTE method.");
run;
```

Close the HTML destination. It is a best practice to reopen the HTML destination after you have closed all of the open destinations.

```sas
  ods html close;
  ods html; /* Not required in SAS Studio */
```

Output 11.3 Adding Text and a Note

Using the DATA Argument

This text is displayed with the FORMAT_TEXT method.

This text is displayed with the NOTE method.

Using the STYLE_ELEM and INLINE_ELEM Arguments to Format Data

```sas
filename rwiOut ".";
ods html close;
ods html path=rwiOut file="StyleElemText.html";
  title "Using the STYLE_ELEM and INLINE_ELEM Arguments";
  title "Using the STYLE_ELEM and INLINE_ELEM Arguments";
data _null_
  dcl odsout obj();
  obj.format_text(data: "This text is formatted using the STYLE_ELEM argument and the SystemTitle Style Element. ",
```
Program Description

Create a file reference for the output, set the ODS options, and specify a title. The current working directory is specified in this example.

```sas
filename rwiOut ".";
ods html close;
ods html path=rwiOut file="StyleElemText.html"
  title "Using the STYLE_ELEM and INLINE_ELEM Arguments";
```

Begin the DATA Null step and declare the local variable obj. The DATA _NULL_ statement begins the DATA step. The DECLARE statement assigns obj as the variable that identifies the object. The DECLARE statement also creates an instance.

```sas
title "Using the STYLE_ELEM and INLINE_ELEM Arguments";
data _null_;  
dcl odsout obj();
```

Apply the style element SystemTitle to all of the text specified by this FORMAT_TEXT method. The FORMAT_TEXT method with the DATA argument specified adds text to all open destinations. The STYLE_ELEM argument specifies that the style element SystemTitle is used to format the text. The SystemTitle style element applies to all of the text specified in this method, no matter where STYLE_ELEM is specified within the method.

```sas
obj.format_text(data: "This text is formatted using the STYLE_ELEM argument and the SystemTitle Style Element. ",
  style_elem: "SystemTitle",
  data: "The STYLE_ELEM argument applies to the entire FORMAT_TEXT method.");
```

```sas
obj.note(data: "This note is formatted using the STYLE_ELEM argument and the SystemTitle Style Element. ",
  style_elem: "SystemTitle",
  data: "The STYLE_ELEM argument applies to the entire NOTE method.");
```

```sas
obj.note(data: "This note is formatted using the INLINE_ELEM argument and the SystemTitle Style Element. ",
  inline_elem: "SystemTitle",
  data: "The INLINE_ELEM argument applies only to the preceding text.");
```

```sas
run;
ods html close;
```

/* Not required in SAS Studio */
Apply the style element SystemTitle to part of the text specified by this FORMAT_TEXT method. The FORMAT_TEXT method with the DATA argument specified adds text to all open destinations. The INLINE_ELEM argument specifies that the style element SystemTitle is used to format the text. The SystemTitle style element applies only to the text specified by the DATA argument that directly precedes the INLINE_ELEM argument.

```sas
obj.format_text(data: "This text is formatted using the INLINE_ELEM argument and the SystemTitle Style Element.",
    inline_elem: "SystemTitle",
    data: "The INLINE_ELEM argument applies only to the preceding text.");
```

Apply the style element SystemTitle to all of the text specified by this NOTE method. The NOTE method with the DATA argument specified adds a note to all open destinations. The STYLE_ELEM argument specifies that the style element SystemTitle is used to format the text. The SystemTitle style element applies to all of the text specified in this method, no matter where STYLE_ELEM is specified within the method.

```sas
obj.note(data: "This note is formatted using the STYLE_ELEM argument and the SystemTitle Style Element.",
    style_elem: "SystemTitle",
    data: "The STYLE_ELEM argument applies to the entire NOTE method.");
```

Apply the style element SystemTitle to part of the text specified by this NOTE method. The NOTE method with the DATA argument specified adds a note to all open destinations. The INLINE_ELEM argument specifies that the style element SystemTitle is used to format the text. The SystemTitle style element applies only to the text specified by the DATA argument that directly precedes the INLINE_ELEM argument.

```sas
obj.note(data: "This note is formatted using the INLINE_ELEM argument and the SystemTitle Style Element.",
    inline_elem: "SystemTitle",
    data: "The INLINE_ELEM argument applies only to the preceding text.");
```

run;

Close the HTML destination. It is a best practice to reopen the HTML destination after you have closed all of the open destinations.

```sas
ods html close;
ods html; /* Not required in SAS Studio */
```

Output 11.4  Adding the STYLE_ELEM and INLINE_ELEM Arguments
Using the STYLE_ATTR and INLINE_ATTR Arguments to Format Data

filename rwiOut ".";
ods html close;
ods html path=rwiOut file="StyleAttrText.html";
title "Using the INLINE_ATTR and STYLE_ATTR Arguments";
data _null_
dcl odsout obj();

obj.format_text(data: "With the STYLE_ATTR argument, ",
                 style_attr: "color=red",
                 data: "the style attribute applies to all of the text.");

obj.note(data: "With the INLINE_ATTR argument, you could make this text red, ",
           inline_attr: "color=red",
           data: "and this text bold, ",
           inline_attr: "font_weight=bold",
           data: "and use a 16pt font. ",
           inline_attr: "font_size=16pt");
run;
ods html close;
ods html; /* Not required in SAS Studio */

Program Description

Create a file reference for the output, set the ODS options, and specify a title. The current working directory is specified in this example.

filename rwiOut ".";
ods html close;
ods html path=rwiOut file="StyleAttrText.html";
title "Using the INLINE_ATTR and STYLE_ATTR Arguments";

Begin the DATA Null step and declare the local variable obj. The DATA _NULL_ statement begins the DATA step. The DECLARE statement assigns obj as the variable that identifies the object. The DECLARE statement also creates an instance.

title "Using the INLINE_ATTR and STYLE_ATTR Arguments";
data _null_
dcl odsout obj();

Format all of the text. The FORMAT_TEXT method with the DATA argument specified adds text to all open destinations. The STYLE_ATTR argument specifies that the following style attributes are used to format the text. The style attributes apply to all of the text specified in this method, no matter where STYLE_ATTR is specified within the method.

obj.format_text(data: "With the STYLE_ATTR argument, ",
                 style_attr: "color=red",
                 data: "the style attribute applies to all of the text.");

Example 1: Formatting Data with Style Attributes and Style Elements
Format different parts of the text. The NOTE method with the DATA argument specified adds a note to all open destinations. The INLINE_ATTR argument specifies that the following style attributes are used to format the text. The style attributes apply only to the preceding DATA argument value.

```sas
obj.note(data: "With the INLINE_ATTR argument, you could make this text red, ",
    inline_attr: "color=red",
    data: "and this text bold, ",
    inline_attr: "font_weight=bold",
    data: "and use a 16pt font. ",
    inline_attr: "font_size=16pt";
run;
```

Close the HTML destination. It is a best practice to reopen the HTML destination after you have closed all of the open destinations.

```sas
ods html close;
ods html; /* Not required in SAS Studio */
```

**Output 11.5  Adding the STYLE_ATTR and INLINE_ATTR Arguments**

Using the INLINE_ATTR and STYLE_ATTR Arguments

With the STYLE_ATTR argument, the style attribute applies to all of the text.

With the INLINE_ATTR argument, you could make this text red, and this text bold, and use a 16pt font.

---

**Example 2: Adding Audio to HTML5 Output**

**Features:**
DECLARE statement

**Methods:**
- FOOTNOTE
- FORMAT_TEXT
- NOTE
- AUDIO

**Text arguments:**
- DATA
- STYLE_ATTR
- INLINE_ATTR

**Audio arguments:**
- AUTOPLAY
- FILE
- LOOP
- PRELOAD
- TYPE

**Details**
This example demonstrates the following:
- using the AUDIO method to add an audio file to your output
• using the FORMAT_TEXT method to add text to your output
• using the NOTE method to add a note to your output
• using the FOOTNOTE method to add a footnote to your output
• using style elements to format your text
• using style attributes to format your text

Program

filename rwiOut ".*;
ods html close;
ods html5 path=rwiOut file="AudioTest.html";
title "Adding Audio to HTML5 Output";
data _null_;  
dcl odsout obj();
  obj.format_text(data: "You can embed Video and Audio into your output.",
                  style_elem: "SystemTitle");
filename AudFile url "file-path\SAS02_Orchestral30.mp3";
  obj.note(data: "However, only in the ",
            inline_attr: "font_weight=bold",
            data: "ODS HTML5 ",
            inline_attr: "color=red",
            data: "Destination.",
            inline_attr: "font_weight=bold");

  obj.audio(file:"fileref:AudFile",
           type: "mp3",
           preload: "auto",
           autoplay: "off",
           loop: "no" );

  obj.footnote(data: 'Browsers support different file types.  
                   This output is viewed in Google Chrome.',
                style_attr: 'just=center  color=red  fontsize=12pt');
run;
ods html5 close;
ods html; /* Not required in SAS Studio */

Program Description

Create a file reference for the output. The current working directory is specified in this example.
filename rwiOut ".*;

Open the HTML5 destination and specify a title. The HTML destination is open by default. The ODS HTML CLOSE statement closes the HTML destination to conserve resources.
Begin the DATA Null step and declare the local variable obj. The DATA _NULL_ statement begins the DATA step. The DECLARE statement assigns obj as the variable that identifies the object. The DECLARE statement also creates an instance.

```sql
data _null_
  dcl odsout obj();
```

Add formatted text. The FORMAT_TEXT method with the DATA argument specified adds text to all open destinations. The STYLE_ELEM argument specifies that the SystemTitle style element is used to format the text.

```sql
obj.format_text(data: "You can embed Video and Audio into your output.",
  style_elem: "SystemTitle");
```

Assign a fileref to your audio file. The FILENAME statements assigns a fileref to identify the location of an audio file.

```sql
filename AudFile url "file-path\SAS02_Orchestral30.mp3";
```

Add a note. The NOTE method with the DATA argument specified adds a note to all open destinations. The INLINE_ATTR argument specifies that the following style attributes are used to format the text. The style attributes apply only to the preceding DATA argument value.

```sql
obj.note(data: "However, only in the ",
  inline_attr: "font_weight=bold",
  data: "ODS HTML5 ",
  inline_attr: "color=red",
  data: "Destination.",
  inline_attr: "font_weight=bold");
```

Add an audio file to your output using a fileref. You must use the fileref as the value of the FILEREF identifier. The audio arguments that follow control the behavior of the audio file.

```sql
obj.audio(file: "fileref: AudFile",
  type: "mp3",
  preload: "auto",
  autoplay: "off",
  loop: "no" );
```

Add a footnote. The FOOTNOTE method with the DATA argument specified adds a note to all open destinations. The STYLE_ATTR argument specifies that the following style attributes are used to format the text. The style attributes apply to all of the text specified in this method, no matter where STYLE_ATTR is specified within the method.

```sql
obj.footnote(data: 'Browsers support different file types. This output is viewed in Google Chrome.',
  style_attr: 'just=center color=red fontsize=12pt');
run;
```
Close the HTML5 destination. It is a best practice to reopen the HTML destination after you have closed all of the open destinations.

```sas
ods html5 close;
ods html; /* Not required in SAS Studio */
```

**Output 11.6  Adding Audio to RWI Output**

---

**Adding Audio to HTML5 Output**

You can embed Video and Audio into your output.

*Note: However, only in the ODS HTML5 Destination.*

---

Browsers support different file types. This output is viewed in Google Chrome.

---

**Example 3: Adding a Video to HTML5 Output**

**Features:**
- DECLARE statement

**Methods**
- FOOTNOTE
- FORMAT_TEXT
- NOTE
- VIDEO

**Text arguments**
- DATA
- STYLE_ATTR
- INLINE_ATTR

**Video arguments**
- AUTOPLAY
- FILE
- HEIGHT
- LOOP
- POSTER
- PRELOAD
- TYPE
- WIDTH

**Details**

This example demonstrates the following:

- using the VIDEO method to add a video to your output
- using the FORMAT_TEXT method to add text to your output
- using the NOTE method to add a note to your output
- using the FOOTNOTE method to add a footnote to your output
• using style elements to format your text
• using style attributes to format your text

Program

filename rwiOut "c:\temp";
ods html close;
ods html5 path=rwiOut file="your-file.html";
title "Adding a Video to HTML5 Output";
data _null_;
dcl odsout obj();
  obj.format_text(data: "You can embed Video and Audio into your output.",
                  style_elem: "SystemTitle");
  obj.note(data: "However, only in the ",
            inline_attr: "font_weight=bold",
            data: "ODS HTML5 ",
            inline_attr: "color=red",
            data: "Destination.",
            inline_attr: "font_weight=bold");
            2012/11598_Overview_Final_SASCOM640x360.mp4",
            type: "mp4",
            width: "640",
            height: "480",
            preload: "auto",
            autoplay: "yes",
            poster: "//www.sas.com/content/dam/SAS/en_us/image/
                   sas-com/company/sas-the-power-to-know.jpg",
            loop: "on" );
  obj.footnote(data: 'Browsers support different file types.',
               style_attr: 'just=center color=red fontsize=12pt');
run;
ods html5 close;
ods html; /* Not required in SAS Studio */

Program Description

Create a file reference for the output. The temp directory is specified in this example. You can change the path to any output directory you choose.

filename rwiOut "c:\temp";

Open the HTML5 destination and specify a title. The HTML destination is open by default. The ODS HTML CLOSE statement closes the HTML destination to conserve resources.

ods html close;
ods html5 path=rwiOut file="your-file.html";
title "Adding a Video to HTML5 Output";
Begin the DATA Null step and declare the local variable `obj`. The DATA _NULL_ statement begins the DATA step. The DECLARE statement assigns obj as the variable that identifies the object. The DECLARE statement also creates an instance.

```
data _null_;  
dcl odsout obj();
```

**Add formatted text.** The FORMAT_TEXT method with the DATA argument specified adds text to all open destinations. The STYLE_ELEM argument specifies that the SystemTitle style element is used to format the text.

```
obj.format_text(data: "You can embed Video and Audio into your output.",  
                style_elem: "SystemTitle");
```

**Add a note.** The NOTE method with the DATA argument specified adds a note to all open destinations. The INLINE_ATTR argument specifies that the following style attributes are used to format the text. The style attributes apply only to the preceding DATA argument value.

```
obj.note(data: "However, only in the ",  
         inline_attr: "font_weight=bold",  
         data: "ODS HTML5 ",  
         inline_attr: "color=red",  
         data: "Destination.",  
         inline_attr: "font_weight=bold");
```

**Add the video file.** The VIDEO method adds the video file that is specified by the FILE: argument. The video arguments that follow control the behavior and look of the video file.

```
         2012/11598_Overview_Final_SASCOM640x360.mp4",  
          type: "mp4",  
          width: "640",  
          height: "480",  
          preload: "auto",  
          autoplay: "yes",  
          poster: "/www.sas.com/content/dam/SAS/en_us/image/  
               sas-com/company/sas-the-power-to-know.jpg",  
          loop: "on"  
         );
```

**Add a footnote.** The FOOTNOTE method with the DATA argument specified adds a note to all open destinations. The STYLE_ATTR argument specifies that the following style attributes are used to format the text. The style attributes apply to all of the text specified in this method, no matter where STYLE_ATTR is specified within the method.

```
obj.footnote(data: 'Browsers support different file types.',  
             style_attr: 'just=center color=red fontsize=12pt');
```

```
run;
```

**Close the HTML5 destination.** It is a best practice to reopen the HTML destination after you have closed all of the open destinations.

```
ods html5 close;  
ods html; /* Not required in SAS Studio */
```
Output 11.7  Adding Video to RWI Output

Adding a Video to HTML5 Output
You can embed Video and Audio into your output.

Note: However, only in the ODS HTML5 Destination.

Example 4: Creating a Flyer with Text Methods

Features: DECLARE statement  
Text Methods  
  FORMAT_TEXT  
  NOTE

Other features: FOOTNOTE statement  
  IF-THEN statement  
  ODS ESCAPECHAR statement  
  ODS PDF statement  
  TITLE statement
Details

The following example creates a company flyer using text methods only. You can use text methods to add text, images, and notes. You can also customize your text with style attributes and organize your text with IF-THEN processing.

Program

```sas
ods html close;
options nodate nonumber;
ods escapechar="~";
options papersize=(6in 4in);

data brand;
    length tagline $24 description $128;
    input tagline $24. / description $128.;
datalines;
Our Mission
To deliver quality sporting and outdoor equipment for all seasons at the most affordable prices.
Our Vision
To transform the way the world purchases sporting and outdoor equipment.
Our Values
Customer focused, Swift and Agile, Innovative, Trustworthy
Our Goal
To grow sales by 15% annually while also improving profit margin through innovative thinking.
;
run;

ods pdf file="text.pdf" notoc;
title "~-{style [width=100pct background=cx494068
color=cxbbb2e0 font_size=24pt just=left] Our Company}~";
footnote "~-{style [font_size=10pt color=cxbbb2e0]
Created with the ODS Report Writing Interface.}~";

data _null_; 
    set brand end=eof;
    if _n_ = 1 then do;
        declare odsout obj();
        obj.note(data: "Our Brand...",
            style_attr: "font_style=italic
            font_size=12pt
            font_weight=bold
            color=cxbbb2e0",
            just: "L");
    end;

    obj.format_text(data: tagline,
        style_attr: "background=cx494068
        color=cxbbb2e0
        font_size=12pt
        font_style=italic
        width=2.5in",
```
Program Description

Set the SAS system options and define a representative character to be used with inline styles. The ODS HTML destination is open by default. If you are not creating HTML output, close the HTML destination to conserve system resources.

```sas
ods html close;
options nodate nonumber;
ods escapechar="~";
```

Create the input data set.

```sas
data brand;
  length tagline $24 description $128;
  input tagline $24. / description $128.;
datalines;
Our Mission
To deliver quality sporting and outdoor equipment for all seasons at the most affordable prices.
Our Vision
To transform the way the world purchases sporting and outdoor equipment.
Our Values
Customer focused, Swift and Agile, Innovative, Trustworthy
Our Goal
To grow sales by 15% annually while also improving profit margin through innovative thinking.
;
run;
```

Open the PDF destination and specify titles and footnotes using inline styles.

```sas
ods pdf file="text.pdf" notoc;
title "-{style [width=100pct background=cx494068
  color=cxbbb2e0 font_size=24pt just=left] Our Company}";
footnote "-{style [font_size=10pt color=cxbbb2e0]
  Created with the ODS Report Writing Interface.}";
```
Begin the DATA step and specify the actions to execute on the first observation. The statements and methods between the following IF-THEN statement and the END statement are executed only once on the first observation. The local object variable obj needs to be initialized only once for the duration of the DATA step. The DCL statement declares the ODS object and assigns it to the local variable obj. The DCL statement also creates a run-time instance of obj. The NOTE method adds an image and text to the top of the output.

```plaintext
data _null_;  
set brand end=eof; 

if _n_ = 1 then do;  
declare odsout obj();  
obj.note(data: "Our Brand...",  
  style_attr: "font_style=italic  
  font_size=12pt  
  font_weight=bold  
  color=cxbbb2e0",  
  just: "L");  
end; 
```

Add the input data. The FORMAT_TEXT method adds and formats the text. The FORMAT_TEXT methods are not surrounded by any conditional logic, so they are executed for each observation in the input data set. The values specified by the DATA argument are not in quotation marks. They are used to reference variable names in the input data set. The values of these variables are displayed in the output.

```plaintext
obj.format_text(data: tagline,  
  style_attr: *background=cx494068  
  color=cxbbb2e0  
  font_size=12pt  
  font_style=italic  
  width=2.5in",  
  just: "C");

obj.format_text(data: description,  
  style_attr: *background=cxbbb2e0  
  font_style=italic  
  font_size=8pt  
  width=2.5in",  
  just: "C");
```

Use the FORMAT_TEXT method with IF-THEN processing. This FORMAT_TEXT method adds a blank line at the bottom of your output, and is executed for all but the last observation. By specifying the FORMAT_TEXT method within the IF THEN statement block, the method is executed for all but the last observation.

```plaintext
if eof ne 1 then  
obj.format_text(data: " ",  
  style_attr: *height=1mm");
run;
```
Close all open destinations and open the HTML destination. It is a best practice to reopen the HTML destination after you have closed all of the open destinations.

```sas
ods _all_ close;
ods html; /* Not required in SAS Studio */
```

Output 11.8  Flyer Created with the Text Methods

---

Example 5: Working with Data Sets

**Features:**
- DECLARE statement

**Methods**
- FORMAT_TEXT
- NOTE

**Method Arguments**
- DATA
- STYLE_ELEM
- STYLE_ATTR
- INLINE_ELEM
- INLINE_ATTR

**Details**

This example demonstrates the following:

- adding a text string to your output
- using a variable name to add data from an input data set to your output

**Program: Using the FORMAT_TEXT Method with a Text String**

```sas
filename rwiOut ".";
ods html close;
```
Program Description

Create a file reference for the output, set the ODS options, and specify a title. The current working directory is specified in this example.

```sas
filename rwiOut ".";
ods html close;
ods html path=rwiOut file="TextString.html";
title "Using the FORMAT_TEXT and NOTE Methods to Display Text";
```

Begin the DATA Null step and declare the local variable obj. The DATA _NULL_ statement begins the DATA step. The DECLARE statement assigns obj as the variable that identifies the object. The DECLARE statement also creates an instance. The SET statement reads the observations from the Sashelp.Class data set.

```sas
title "Using the FORMAT_TEXT and NOTE Methods to Display Text";
data _null_;  
set sashelp.class ;  
dcl odsout obj();
```

Display text. The FORMAT_TEXT method with the DATA argument specified adds text to all open destinations. The NOTE method with the DATA argument specified adds a note to all open destinations.

```sas
obj.format_text(data: "Text Displayed with the FORMAT_TEXT method and the DATA argument.");
obj.note(data: "Text Displayed  with the NOTE method and the DATA argument.");
run;
```

Close the HTML destination. It is a best practice to reopen the HTML destination after you have closed all of the open destinations.

```sas
ods html close;
ods html; /* Not required in SAS Studio */
```

The FORMAT_TEXT and NOTE methods are executed once for each observation in the Sashelp.Class data set. Because text is specified with the DATA argument, the text is written out once for each observation, and no data values are displayed.
Program: Using the FORMAT_TEXT Method with a Variable Name

```plaintext
filename rwiOut ".";
ods html close;
ods html path=rwiOut file="VarText.html";
title "Using a Variable Name to Display Data";
title "Using a Variable Name to Display Data";
data _null_;  
   set sashelp.class ;
dcl odsout obj();

   obj.format_text(data: name, data: weight);

run;
ods html close;
ods html; /* Not required in SAS Studio */
```

Program Description

Create a file reference for the output, set the ODS options, and specify a title. The current working directory is specified in this example.

Begin the DATA Null step and declare the local variable `obj`. The DATA _NULL_ statement begins the DATA step. The DECLARE statement assigns `obj` as the variable
that identifies the object. The DECLARE statement also creates an instance. The SET statement reads the observations from the Sashelp.Class data set.

```sas
title "Using a Variable Name to Display Data";
data _null_;   
set sashelp.class ;
dcl odsout obj();
```

**Specify the DATA set variables to use.** Each DATA argument in the FORMAT_TEXT method specifies a variable from the Sashelp.Class data set. The FORMAT_TEXT method is executed once for each iteration of the data set and the values of the variables Name and Weight are written to the output.

```sas
obj.format_text(data: name, data: weight);
run;
```

**Close the HTML destination.** It is a best practice to reopen the HTML destination after you have closed all of the open destinations.

```sas
ods html close;   
ods html; /* Not required in SAS Studio */
```

**Using a Variable Name to Add Data**

When you use a variable name with the FORMAT_TEXT or NOTE methods, the variable value for each observation is added to your output.

```
Using a Variable Name to Display Data

Alfred 112.5
Alice 84
Barbara 98
Carol 102.5
Henry 102.5
James 83
Jane 84.5
Janet 112.5
Jeffrey 84
John 98.5
Joyce 90.5
Judy 50
Louise 77
Mary 112
Philip 150
Robert 123
Ronald 133
Thomas 85
William 112
```

**Example 6: Working with BY Groups and IF-THEN Processing**

**Features:**
- DECLARE statement

**Methods**
Arguments

DATA
STYLE_ELEM
STYLE_ATTR
INLINE_ELEM
INLINE_ATTR

Other features: IF-THEN statement
BY statement

Details

This example demonstrates the following:

• grouping data with BY groups
• using IF-THEN processing to organize your output
• adding a note to your output

Program

filename rwiOut ".";
ods html close;
ods html path=rwiOut file="ByEx.html";
title "Using BY Groups and IF-THEN Processing";
title "Using BY Groups and IF-THEN Processing";
proc sort data=sashelp.class out=class;
  by sex;
run;
data _null_
  set work.class ;
  by sex;
  if _N_ = 1 then do;
    dcl odsout obj();
  end;
  if first.sex then do;
    obj.format_text(data: sex);
    obj.format_text(data: "This text is added for each first observation
of a By Group using IF-THEN Processing ",
inline_attr: *font_weight=bold*,
data: "The INLINE_ATTR argument is used to
format part of the text.");
  end;
  obj.format_text(data: name);
  if last.sex then do;
    obj.note(data: "This note is added at the end of each BY group with
the NOTE method and IF_THEN Processing. ");
  end;
run;
ods html close;
ods html; /* Not required in SAS Studio */
Program Description

Create a file reference for the output, set the ODS options, and specify a title. The current working directory is specified in this example.

```
filename rwiOut ".";
ods html close;
ods html path=rwiOut file="ByEx.html";
title "Using BY Groups and IF-THEN Processing";
```

Sort the data set. PROC SORT sorts the data by the variable Sex.

```
title "Using BY Groups and IF-THEN Processing";
proc sort data=sashelp.class out=class;
  by sex;
run;
```

Specify the BY group. The BY statement specifies the BY variable.

```
data _null_; 
  set work.class ;
  by sex;
```

Declare and instantiate the variable obj. Because the body of the DATA step is executed once for each input observation during the execution phase, the local object variable obj should be initialized only once for the duration of the DATA step. This can be done using _N_ as follows.

```
if _N_ = 1 then do;
  dcl odsout obj();
end;
```

Use IF-THEN processing to specify the placement of data. To add text to the top of each BY group, use the IF-THEN statement to specify that the two FORMAT_TEXT methods execute only once, for the first observation of each BY group. The INLINE_ATTR argument formats the preceding text.

```
if first.sex then do;
  obj.format_text(data: sex);
  obj.format_text(data: "This text is added for each first observation of a By Group using IF-THEN Processing ",
                  inline_attr: "font_weight=bold",
                  data: "The INLINE_ATTR argument is used to format part of the text.");
end;
```

Specify the character variable to write data from. this line is executed once for each observation in the BY group

```
obj.format_text(data: name);
```

Use the NOTE method with IF-THEN processing to add a note to the end of each BY group. Use the IF THEN statement to specify that the NOTE method executes once, after the last observation in each BY group.

```
if last.sex then do;
  obj.note(data: "This note is added at the end of each BY group with the NOTE method and IF_THEN Processing. ");
end;
```
Close the HTML destination. It is a best practice to reopen the HTML destination after you have closed all of the open destinations.

```
ods html close;
ods html; /* Not required in SAS Studio */
```

### Output 11.10  Using BY Groups and IF-THEN Processing

Using BY-Groups and IF-THEN Processing

F
This text is added for each first observation of a By-Group using IF-THEN Processing
The INLINE_ATTR argument is used to format part of the text.
Alice
Barbara
Carol
Jane
Janet
Joyce
Judy
Louise
Mary

This note is added at the end of each BY-group with the NOTE method and IF_THEN Processing.

M
This text is added for each first observation of a By-Group using IF-THEN Processing
The INLINE_ATTR argument is used to format part of the text.
Alfred
Henry
James
Jeffrey
John
Philip
Robert
Ronald
Thomas
William

This note is added at the end of each BY-group with the NOTE method and IF_THEN Processing.

### Example 7: Creating a Simple Table

**Features:**  DECLARE statement

**Methods**

- FORMAT_CELL
- ROW_END
- ROW_START
- TABLE_END
- TABLE_START

**Method Arguments**

- DATA
- STYLE_ATTR
Details

This example demonstrates the following:

- creating a simple table with an input data set
- formatting cells with style attributes
- using IF-THEN processing to organize your output

Program

```sas
filename rwiOut ".";
ods html close;
ods html path=rwiOut file="ByEx.html";
title "Cars By Make and Model";

data _null_;
  set sashelp.cars end=eof;
  if _n_ = 1 then do;
    dcl odsout obj();
    obj.table_start();
  end;
  obj.row_start();
  obj.format_cell(data: make, style_attr: "color=red");
  obj.format_cell(data: model, style_attr: "color=blue");
  obj.row_end();
  if eof then do;
    obj.table_end();
  end;
run;
ods html close;
ods html; /* Not required in SAS Studio */
```

Program Description

Create a file reference for the output, set the ODS options, and specify a title. The current working directory is specified in this example.

```sas
filename rwiOut ".";
ods html close;
ods html path=rwiOut file="ByEx.html";
title "Cars By Make and Model";
```

Begin the DATA step and specify the actions to execute on the first observation. The statements and methods between the following IF-THEN statement and the END statement are executed only once on the first iteration of the DATA step. The local object variable obj needs to be initialized only once for the duration of the DATA step. The DCL statement declares the ODS object and assigns it to the local variable obj. The DCL statement also creates a run-time instance of obj. The TABLE_START method begins...
Create a row. The ROW_START method creates a table row. The FORMAT_CELL method adds data to the cells in the row. The DATA argument with a variable specified writes out the variable value of Make and Model for each observation. The STYLE_ATTR argument specifies that the data uses the Color style attribute. The ROW_START method and the FORMAT_CELL method are executed once for each observation. For each observation, a new row with two cells is added to the table.

```sas
    obj.row_start();
    obj.format_cell(data: make, style_attr: "color=red");
    obj.format_cell(data: model, style_attr: "color=blue");
    obj.row_end();
```

End the table. The last IF-THEN processing loop specifies that the TABLE_END method is executed after the last observation adds a row to table. The TABLE_END method ends the table.

```sas
    if eof then do;
        obj.table_end();
    end;
run;
```

Close and reopen the HTML destination. It is a best practice to reopen the HTML destination after you have closed all of the open destinations.

```sas
    ods html close;
    ods html; /* Not required in SAS Studio */
```
Example 8: Formatting Cells Using the TextDecoration Style Attribute

**Features:**
- DECLARE statement

**Methods:**
- FORMAT_CELL
- ROW_END
- ROW_START
- TABLE_END
- TABLE_START

**Method Attributes:**
- DATA
- LABEL
- STYLE_ATTR

**Other features:**
- IF-THEN statement
- ODS HTML statement
- ODS PDF statement
- SET statement

---

### Listing of Cars By Manufacturer and Model

<table>
<thead>
<tr>
<th>Manufacturer</th>
<th>Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Acura</td>
<td>MDX</td>
</tr>
<tr>
<td>Acura</td>
<td>RSX Type S 2dr</td>
</tr>
<tr>
<td>Acura</td>
<td>TSX 4dr</td>
</tr>
<tr>
<td>Acura</td>
<td>TL 4dr</td>
</tr>
<tr>
<td>Acura</td>
<td>3.5 RL 4dr</td>
</tr>
<tr>
<td>Acura</td>
<td>3.5 RL w/Navigation 4dr</td>
</tr>
<tr>
<td>Acura</td>
<td>NSX coupe 2dr manual S</td>
</tr>
<tr>
<td>Audi</td>
<td>A4 1.8T 4dr</td>
</tr>
<tr>
<td>Audi</td>
<td>A4 1.8T convertible 2dr</td>
</tr>
<tr>
<td>Audi</td>
<td>A4 3.0 4dr</td>
</tr>
<tr>
<td>Audi</td>
<td>A4 3.0 Quattro 4dr manual</td>
</tr>
<tr>
<td>Audi</td>
<td>A4 3.0 Quattro 4dr auto</td>
</tr>
<tr>
<td>Audi</td>
<td>A6 3.0 Quattro 4dr</td>
</tr>
<tr>
<td>Audi</td>
<td>A4 3.0 convertible 2dr</td>
</tr>
<tr>
<td>Audi</td>
<td>A4 3.0 Quattro convertible 2dr</td>
</tr>
<tr>
<td>Audi</td>
<td>A6 2.7 Turbo Quattro 4dr</td>
</tr>
<tr>
<td>Audi</td>
<td>A6 4.2 Quattro 4dr</td>
</tr>
<tr>
<td>Audi</td>
<td>A8 L Quattro 4dr</td>
</tr>
<tr>
<td>Audi</td>
<td>S4 Quattro 4dr</td>
</tr>
</tbody>
</table>
Details
This example demonstrates the following:

- creating a table with the TABLE_START and ROW_START methods
- formatting cells with the FORMAT_CELL method
- adding lines over, under, and through text with the TEXTDECORATION style attribute

The Textdecoration style attribute does not appear the same in all of the destinations.

Program

```sas
filename rwiOut ".";
ods html close;
title "Using the TextDecoration Style Attribute";

options nodate nonumber;
ods html path=rwiOut file="textdecoration.html";
ods pdf file="your-path/textdecoration.pdf";
title "Using the TextDecoration Style Attribute";

data _null_;  
  set sashelp.class end=eof;
  if _n_ = 1 then do;
    dcl odsout obj();
    obj.table_start(label: "Text Decoration Test") ;
  end;
  obj.row_start();
    obj.format_cell(data: age, style_attr: "textdecoration=overline");
    obj.format_cell(data: height, style_attr: "textdecoration=underline");
    obj.format_cell(data: weight, style_attr: "textdecoration=line_through");
    obj.format_cell(data: name);
  obj.row_end();
  if eof then do;
    obj.table_end();
  end;
run;

ods _all_ close;
ods html; /* Not required in SAS Studio */
```

Program Description

**Create a file reference for the output and specify a title.** The current working directory is specified in this example.

```sas
filename rwiOut ".";
ods html close;
title "Using the TextDecoration Style Attribute";
```

**Specify the ODS destinations and filenames for the output.**
options nodate nonumber;
ods html path=rwiOut file="textdecoration.html";
ods pdf file="your-path/textdecoration.pdf";
title "Using the TextDecoration Style Attribute";

Begin the DATA step and specify the actions to execute on the first iteration. The statements and methods between the following IF-THEN statement and the END statement are executed only once on the first iteration of the DATA step. The local object variable obj needs to be initialized only once for the duration of the DATA step. The DCL statement declares the ODS object and assigns it to the local variable obj. The DCL statement also creates a run-time instance of obj. The TABLE_START method begins the table. The LABEL argument specifies the label of that will be used in the table of contents and the DMS Results window. The DCL statement and the TABLE_START method are executed once on the first iteration of the DATA step.

data _null_;  
set sashelp.class end=eof;  
if _n_ = 1 then do;  
dcl odsout obj();  
   obj.table_start(label: "Text Decoration Test" );  
end;

Create rows. The ROW_START method creates a table row. The FORMAT_CELL method adds data to the cells in the row. The DATA argument with a variable specified writes out the variable value of each observation. The STYLE_ATTR argument specifies that the data use the TEXTDEcoration style attribute. The ROW_START – ROW_END method block is executed once for each observation. For each observation, a new row with four cells is added to the table.

obj.row_start();  
obj.format_cell(data: age,  
   style_attr: "textdecoration=overline" );  
obj.format_cell(data: height,  
   style_attr: "textdecoration=underline" );  
obj.format_cell(data: weight,  
   style_attr: "textdecoration=line_through" );  
obj.format_cell(data: name);  
obj.row_end();

End the table. The last IF-THEN processing loop specifies that the TABLE_END method is executed after the last observation. The TABLE_END method ends the table.

if eof then do;  
   obj.table_end();  
end;  
run;

Close the open destinations. It is a best practice to reopen the HTML destination after you have closed all of the open destinations.

ods _all_ close;  
ods html; /* Not required in SAS Studio */
### Using the TextDecoration Style Attribute

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>14</td>
<td>69</td>
<td>442.5</td>
<td>Alfred</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>56.5</td>
<td>84</td>
<td>Alice</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>65.3</td>
<td>96</td>
<td>Barbara</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>62.8</td>
<td>402.5</td>
<td>Carol</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>63.5</td>
<td>402.5</td>
<td>Henry</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>57.3</td>
<td>83</td>
<td>James</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>59.8</td>
<td>425</td>
<td>Jane</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>62.5</td>
<td>412.5</td>
<td>Janet</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>62.5</td>
<td>84</td>
<td>Jeffrey</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>59</td>
<td>99.5</td>
<td>John</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>51.3</td>
<td>60.5</td>
<td>Joyce</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>64.3</td>
<td>90</td>
<td>Judy</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>56.3</td>
<td>112</td>
<td>Louise</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>66.5</td>
<td>412</td>
<td>Mary</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>72</td>
<td>450</td>
<td>Philip</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>64.8</td>
<td>128</td>
<td>Robert</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>67</td>
<td>133</td>
<td>Ronald</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>57.5</td>
<td>85</td>
<td>Thomas</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>66.5</td>
<td>442</td>
<td>William</td>
<td></td>
</tr>
</tbody>
</table>
Example 9: Creating a Table with Row and Column Spanning

Features:
- DECLARE statement
- Table methods
  - FORMAT_CELL
  - HEADER_END
  - HEADER_START
  - ROW_END
  - ROW_START
  - TABLE_END
  - TABLE_START

Other features:
- BY statement
- IF-THEN statement
- PROC FORMAT
- PROC SORT
- SET statement

Details

The following program creates a table with the following features:

- header section
- spanned columns
• spanned rows
• BY groups

Program

options nodate nonumber;
filename rwiOut ".";
ods html close;
ods html path=rwiOut file="MyEx.html";
title "Table Using Column and Row Spanning";
title2 "Two Tables, One for Females and One for Males";
proc format;
  value $gender
    F = "Females"
    M = "Males"
  ;
run;

proc sort data=sashelp.class out=class;
  by sex;
run;
data _null_;  
set class;
  by sex;
if _N_ = 1 then do;
  dcl odsout obj();
end;
if first.sex then do;
  obj.table_start();
  obj.head_start();
  obj.row_start();
    obj.format_cell(data: "Name", row_span: 3, vjust: "B");
    obj.format_cell(data: sex, column_span: 4, format: "$gender.");
  obj.row_end();
  obj.row_start();
    obj.format_cell(data: "Age", row_span: 2, vjust: "B");
    obj.format_cell(data: "Vitals", column_span: 3);
  obj.row_end();
  obj.row_start();
    obj.format_cell(data: "Height");
    obj.format_cell(data: "Weight");
    obj.format_cell(data: "BMI");
  obj.row_end();
  obj.head_end();
end;

bmi = ( weight / ( height * height ) ) * 703;

options nodate nonumber;
filename rwiOut ".";
ods html close;
ods html path=rwiOut file="MyEx.html";
title "Table Using Column and Row Spanning";
title2 "Two Tables, One for Females and One for Males";
proc format;
  value $gender
    F = "Females"
    M = "Males"
  ;
run;

proc sort data=sashelp.class out=class;
  by sex;
run;
data _null_;  
set class;
  by sex;
if _N_ = 1 then do;
  dcl odsout obj();
end;
if first.sex then do;
  obj.table_start();
  obj.head_start();
  obj.row_start();
    obj.format_cell(data: "Name", row_span: 3, vjust: "B");
    obj.format_cell(data: sex, column_span: 4, format: "$gender.");
  obj.row_end();
  obj.row_start();
    obj.format_cell(data: "Age", row_span: 2, vjust: "B");
    obj.format_cell(data: "Vitals", column_span: 3);
  obj.row_end();
  obj.row_start();
    obj.format_cell(data: "Height");
    obj.format_cell(data: "Weight");
    obj.format_cell(data: "BMI");
  obj.row_end();
  obj.head_end();
end;

bmi = ( weight / ( height * height ) ) * 703;

options nodate nonumber;
filename rwiOut ".";
ods html close;
ods html path=rwiOut file="MyEx.html";
title "Table Using Column and Row Spanning";
title2 "Two Tables, One for Females and One for Males";
proc format;
  value $gender
    F = "Females"
    M = "Males"
  ;
run;

proc sort data=sashelp.class out=class;
  by sex;
run;
data _null_;  
set class;
  by sex;
if _N_ = 1 then do;
  dcl odsout obj();
end;
if first.sex then do;
  obj.table_start();
  obj.head_start();
  obj.row_start();
    obj.format_cell(data: "Name", row_span: 3, vjust: "B");
    obj.format_cell(data: sex, column_span: 4, format: "$gender.");
  obj.row_end();
Program Description

Create a file reference for the output, set the ODS options, and specify titles. The current working directory is specified in this example.

```sas
options nodate nonumber;
filename rwiOut ".";
ods html close;
ods html close;
ods html path=rwiOut file="MyEx.html";
title "Table Using Column and Row Spanning";
title2 "Two Tables, One for Females and One for Males";
```

Create a custom format for the BY variable.

```sas
proc format;
  value $gender
    F = "Females"
    M = "Males";
run;
```

Sort the data set. PROC SORT sorts the data by the variable Sex.

```sas
proc sort data=sashelp.class out=class;
  by sex;
run;
```

Begin the DATA step, sort the data set, and specify the actions to execute on the first iteration of the DATA step. The statements and methods between the following IF-THEN statement and the END statement are executed only once on the first iteration. The local object variable obj needs to be initialized only once for the duration of the DATA step. The DCL statement declares the ODS object and assigns it to the local variable obj. The DCL statement also creates a run-time instance of obj.

```sas
data _null_
  set class;
  by sex;

  if _N_ = 1 then do;
    dcl odsout obj();
  end;
```

Begin the table and header section for each BY group. The TABLE_START method begins the table. The TABLE_START method is executed once at the first observation for each BY group. The HEAD_START method begins the header section.
if first.sex then do;
    obj.table_start();
    obj.head_start();

Create the first row of the header. The ROW_START method begins the first row of the header. The FORMAT_CELL methods format the cells. The DATA argument on the first FORMAT_CELL method adds the text ‘Name’ to the first cell of the row, and spans the first three rows. The VJUST argument places the text at the bottom of the cell. The DATA argument on the second FORMAT_CELL argument adds the value of the variable Sex to the second cell in the first row. The value of the variable is used because there are no quotation marks around the variable name. The COLUMN_SPAN argument spans the columns for the variable Sex, and the FORMAT argument applies the custom format $gender. The ROW_END method ends the row.

    obj.row_start();
    obj.format_cell(data: "Name", row_span: 3, vjust: "B");
    obj.format_cell(data: sex, column_span: 4, format: "$gender.");
    obj.row_end();

Create the second row of the header. The ROW_START method begins the second row of the header. The FORMAT_CELL methods format each cell. The first FORMAT_CELL method adds the text ‘Age’ to the second cell. The ROW_SPAN argument spans the cell over two rows. The VJUST argument places the text at the bottom of the cell. The second FORMAT_CELL method add the text ‘Vitals’ to the second cell row. The COLUMN_SPAN argument spans three cells for the text "Vitals". The ROW_END method ends the row.

    obj.row_start();
    obj.format_cell(data: "Age", row_span: 2, vjust: "B");
    obj.format_cell(data: "Vitals", column_span: 3);
    obj.row_end();

Create the third row of the header. The ROW_START method begins the third row of the header. The FORMAT_CELL methods with the DATA argument specified adds the text Height, Weight, and BMI to the third, fourth, and fifth cell. The ROW_END method ends the row.

    obj.row_start();
    obj.format_cell(data: "Height");
    obj.format_cell(data: "Weight");
    obj.format_cell(data: "BMI");
    obj.row_end();

End the header and the IF-THEN block. The HEAD_END method ends the header section of the table. The END statement ends the IF-THEN block. All of the statements between the first IF-THEN statement and the END statement are executed once, for the first iteration of each BY group only.

    obj.head_end();
end;

Calculate the BMI for each observation. The new variable BMI is calculated for every observation in the data set.

    bmi = ( weight / ( height * height ) ) * 703;
Create the body of the table. The block of methods that creates the row is executed once for each observation. The value of the variable is used because there are no quotation marks around the variable name.

```sas
obj.row_start();
obj.format_cell(data: name);
obj.format_cell(data: age);
obj.format_cell(data: height);
obj.format_cell(data: weight);
obj.format_cell(data: bmi, format: "8.2");
obj.row_end();
```

End the table. The TABLE_END method should be executed once for the last observation of each BY group. By specifying the TABLE_END method with the IF LAST THEN statement, the method is executed once on the last observation of the BY group.

```sas
if last.sex then do;
    obj.table_end();
end;
run;
```

Close the HTML destination. It is a best practice to reopen the HTML destination after you have closed all of the open destinations.

```sas
ods html close;
ods html; /* Not required in SAS Studio */
```
Output 11.13  Creating a Table with Cell Spanning and BY Groups

Table Using Column and Row Spanning
Two Tables, One for Females and One for Males

<table>
<thead>
<tr>
<th>Name</th>
<th>Age</th>
<th>Height</th>
<th>Weight</th>
<th>BMI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>13</td>
<td>56.5</td>
<td>94</td>
<td>18.5</td>
</tr>
<tr>
<td>Barbara</td>
<td>13</td>
<td>65.3</td>
<td>90</td>
<td>16.16</td>
</tr>
<tr>
<td>Carol</td>
<td>14</td>
<td>62.8</td>
<td>102.5</td>
<td>18.27</td>
</tr>
<tr>
<td>Jane</td>
<td>12</td>
<td>59.8</td>
<td>84.5</td>
<td>16.61</td>
</tr>
<tr>
<td>Janet</td>
<td>15</td>
<td>62.5</td>
<td>112.5</td>
<td>20.25</td>
</tr>
<tr>
<td>Joyce</td>
<td>11</td>
<td>51.3</td>
<td>50.5</td>
<td>13.49</td>
</tr>
<tr>
<td>Judy</td>
<td>14</td>
<td>64.3</td>
<td>90</td>
<td>15.30</td>
</tr>
<tr>
<td>Louise</td>
<td>12</td>
<td>56.3</td>
<td>77</td>
<td>17.08</td>
</tr>
<tr>
<td>Mary</td>
<td>15</td>
<td>66.5</td>
<td>112</td>
<td>17.80</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>Age</th>
<th>Height</th>
<th>Weight</th>
<th>BMI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alfred</td>
<td>14</td>
<td>69</td>
<td>112.5</td>
<td>16.61</td>
</tr>
<tr>
<td>Henry</td>
<td>14</td>
<td>63.5</td>
<td>102.5</td>
<td>17.87</td>
</tr>
<tr>
<td>James</td>
<td>12</td>
<td>57.3</td>
<td>83</td>
<td>17.77</td>
</tr>
<tr>
<td>Jeffrey</td>
<td>62.5</td>
<td>84</td>
<td>15.12</td>
<td></td>
</tr>
</tbody>
</table>

Example 10: Using the Page Method

Features:  DECLARE statement

Table Methods
- TABLE_END
- TABLE_START
- ROW_END
- ROW_START
- FORMAT_CELL

PAGE Methods
- PAGE

Other features:  BY statement
- IF-THEN statement
- ODS ESCAPECHAR statement
- ODS PDF statement
- PROC SORT
- SET statement
Details
The following program creates the following features:

- output separated into BY groups
- one BY group table per page
- spanned columns

Program
```
options nodate nonumber;
filename rwiOut ".";
ods html close;
ods html path=rwiOut file "Cars.html";
ods pdf file="your-file-path/Cars.pdf";
ods escapechar="^";
proc sort data=sashelp.cars out=cars;
    by make model type;
    where upcase(make) in ("AUDI", "BMW", "VOLKSWAGEN");
run;

data _null_;  
    set cars end=done;
    by make;
    if _N_ = 1 then do;
        declare odsout tbl();
    end;
    if first.make then do;
        tbl.title(data: "Cars by Make and Model");
        tbl.table_start();
        tbl.row_start();
        tbl.format_cell(data: "Car Models made by " || strip(make),
                        style_attr: "just=left font_weight=bold",
                        column_span: 6);
        tbl.row_end();
        tbl.row_start();
        tbl.format_cell(data: " ", inhibit: "LR",
                        column_span: 6);
        tbl.row_end();
        tbl.row_start();
        tbl.format_cell(data: "Model");
        tbl.format_cell(data: "Type");
        tbl.format_cell(data: "HP");
        tbl.format_cell(data: "MPG*(City)", split: "**");
        tbl.format_cell(data: "MPG*(Highway)", split: "**");
        tbl.format_cell(data: "MSRP");
        tbl.row_end();
    end;
```
Program Description

Create a file reference for the HTML output, set the ODS options, and specify titles. The current working directory is specified in this example.

```sas
options nodate nonumber;
filename rwiOut ".";
ods html close;
```

Open the ODS destinations and define a representative character to be used for inline styles.

```sas
ods html path=rwiOut file "Cars.html";
ods pdf file="your-file-path/Cars.pdf";
ods escapechar="^";
```

Sort the input data set by make and model.

```sas
proc sort data=sashelp.cars out=cars;
    by make model type;
    where upcase(make) in ("AUDI","BMW","VOLKSWAGEN");
run;
```

Begin the DATA step and specify the actions to execute on the first iteration of the DATA step. The statements and methods between the following IF-THEN statement and the END statement are executed only once on the first iteration of the DATA step. The local object variable tbl should be initialized only once for the duration of the DATA step. The DCL statement declares the ODS object and assigns it to the local variable tbl. The DCL statement also creates a run-time instance of tbl. It is important to use conditional processing carefully when you are using the PAGE method. The END=DONE option in the SET statement, together with the IF-THEN statement in the last code block, ensures that there is not a blank page with just a title after the last observation. The END option creates and names the temporary variable Done that
contains an end-of-file indicator. The variable, which is initialized to zero, is set to 1 when SET reads the last observation of the last data set listed.

```sas
data _null_;  
  set cars end=done;  
  by make;  
  if _N_ = 1 then do;  
    declare odsout tbl();  
  end;  

Use IF-THEN processing to specify the placement of data. To add text to the top of each BY group, use conditional processing and table methods. The first ROW_START block adds the text "Car Models made by Volkswagen" and spans the columns. The second ROW_START block creates a blank row, and the third ROW_START block adds the headers Model, Type, HP, MPG (City), MPG (Highway), and MSRP. These statements are executed once for the first observation of a BY group only, because of the conditional processing.

```sas
if first.make then do;  
  tbl.title(data: "Cars by Make and Model");  
  tbl.table_start();  
  tbl.row_start();  
  tbl.format_cell(data: "Car Models made by " || strip(make),  
                  style_attr: "just=left font_weight=bold",  
                  column_span: 6);  
  tbl.row_end();  
  tbl.row_start();  
  tbl.format_cell(data: " ", inhibit: "LR",  
                  column_span: 6);  
  tbl.row_end();  
  tbl.row_start();  
  tbl.format_cell(data: "Model");  
  tbl.format_cell(data: "Type");  
  tbl.format_cell(data: "HP");  
  tbl.format_cell(data: "MPG*(City)", split: "*");  
  tbl.format_cell(data: "MPG*(Highway)", split: "*");  
  tbl.format_cell(data: "MSRP");  
  tbl.row_end();  
end;
```

Display the input data. The ROW_START method block creates a row that uses the value of the variable specified by the DATA argument. Because there is no IF-THEN processing, the methods are executed for every observation in the input data set.

```sas
tbl.row_start();  
  tbl.format_cell(data: model);  
  tbl.format_cell(data: type);  
  tbl.format_cell(data: horsepower);  
  tbl.format_cell(data: MPG_city);  
  tbl.format_cell(data: MPG_Highway);  
  tbl.format_cell(data: msrp);  
  tbl.row_end();
```
Use the TABLE END statement with IF-THEN processing to end the table and create a page for each BY group. The TABLE END method ends the table, and the PAGE method forces a new page for each BY group. The IF-NOT DONE statement block specifies that the PAGE method is not executed again after the end of the data set.

\[
\begin{align*}
\text{if last.make then do;}
\text{tbl.table_end();}
\text{if not done then do;}
\text{tbl.page();}
\text{end;}
\text{end;}
\text{run;}
\end{align*}
\]

Close the open destinations. It is a best practice to reopen the HTML destination after you have closed all of the open destinations.

\[
\begin{align*}
\text{ods _all_ close;}
\text{ods html; /* Not required in SAS Studio */}
\end{align*}
\]

Output Using IF-THEN

You should use the PAGE method with additional IF-THEN processing in the SET statement and in the last IF-THEN statement block. When you do this, the PAGE method will not be executed after the end of the data set. There will be no extraneous page after the last page of output.

**Output 11.14  Last Page of PDF Output: No Extra Page**
If you do not use the END= option in the SET statement, along with the IF-THEN statement block at the end with the page method, then there is an extra page with a title after the last full page of output.
Output 11.16  Last Page of PDF Output: Extra Page
Example 11: Using Colors to Understand Absolute Layout

Features:
- DECLARE statement

Absolute Layout Methods
- LAYOUT_ABSOLUTE
- LAYOUT_END
- REGION

Text Method
- FORMAT_TEXT

Other features:
- FOOTNOTE statement
- ODS ESCAPECHAR statement
- ODS PDF statement
- TEMPLATE procedure
Program 1

With absolute layout, the order that your regions are specified is critical. The regions are laid down in the order in which they are specified, beginning with the top of the program. In the following example, region 2 is laid down first, then region 3, then region 4, and last region 5, like pieces of paper. If you do not specify your X and Y options carefully, regions can overlap each other.

For example, in the following code, regions 2 and 3 are overlaid by regions 4 and 5. Some of the red color from region 4 is showing, even though region 5 is specified last. This is because region 4 begins two inches from the top of the page, and region 5 begins seven inches from the top of the page. All of the regions lie on top of the purple background (1), which is specified by the template Orionbackground.

```sas
ods html close;
options nodate nonumber;
title;
ods escapechar="~";
footnote "-{style [font_size=10pt just=right color=cxbbb2e0]
Provided to you by SAS 9.4 and ODS Absolute Layout features.}";
proc template;
define style Styles.Orionbackground;
parent=Styles.Printer;
style body /
  background=cx494068;
end;
run;
ods pdf file="your-file-path/Color3.pdf" style=Styles.Orionbackground notoc;
data _null_;
dcl odsout trt();
trt.layout_absolute();
2  trt.region(y: "5in", style_attr:"backgroundcolor=orange");
   trt.format_text(data: "Executive Prospectus",
                   just: "c",
                   style_attr:"font_size=36pt color=cxbbb2e0");
3  trt.region(y: "3in", x: "3in", style_attr:"backgroundcolor=yellow");
   trt.format_text(data: "Sports & Outdoors",
                   style_attr:"color=cxbbb2e0 font_size=28pt");
4  trt.region(y: "2in", style_attr:"backgroundcolor=red");
   trt.format_text(data: "Orion Star", just: "c",
                   style_attr: "color=cxbbb2e0 just=center font_size=72pt");
5  trt.region(y: "7in", style_attr:"backgroundcolor=blue");
   trt.format_text(data: "For years 1999 through 2002",
                   just: "c",
                   style_attr:"font_size=20pt color=cxbbb2e0");
trt.layout_end();
run;
ods _all_ close;
```
Output 1

**Output 11.18  Regions Overlapping Each Other**

---

**Program 2**

In this program, regions 3 and 4 overlap each other. You can fix this by changing the X and Y values, or by simply changing the order that the regions are specified in.

```sas
ods html close;
options nodate nonumber;
title;
ods escapechar="~";
footnote "-{style [font_size=10pt just=right color=cxbbb2e0] Provided to you by SAS 9.4 and ODS Absolute Layout features.}"
proc template;
  define style Styles.Orionbackground;
  parent=Styles.Printer;
```
style body /
  background=cx494068;
end;
run;
ods pdf file="your-file-path/Color2.pdf" style=Styles.Orionbackground notoc;
data _null_; dcl odsout trt();
trt.layout_absolute();
trt.region(y: "2in", style_attr:"backgroundcolor=red");
  trt.format_text(data: "Orion Star", just: "c",
      style_attr: "color=cxbbb2e0 just=center font_size=72pt");

trt.region(y: "5in", style_attr:"backgroundcolor=orange");
  trt.format_text(data: "Executive Prospectus",
      just: "c",
      style_attr:"font_size=36pt color=cxbbb2e0");

trt.region(y: "3in", x: "3in", style_attr:"backgroundcolor=yellow");
  trt.format_text(data: "Sports & Outdoors",
      style_attr:"color=cxbbb2e0 font_size=28pt");

trt.region(y: "7in", style_attr:"backgroundcolor=blue");
  trt.format_text(data: "For years 1999 through 2002",
      just: "c",
      style_attr:"font_size=20pt color=cxbbb2e0");
trt.layout_end();
run;
ods pdf close;
Program 3

With the regions specified in the correct order, all of the text is visible. You can then remove the region colors for your final output.

```sas
ods html close;
options nodate nonumber;
title;
ods escapechar="~";
footnote "~-{style [font_size=10pt just=right color=cxbbb2e0]
Provided to you by SAS 9.4 and ODS Absolute Layout features.}";
proc template;
  define style Styles.Orionbackground;
    parent=Styles.Printer;
```
style body /
  background=cx494068;
end;
run;
ods pdf file="your-file-path/Color1.pdf" style=Styles.Orionbackground notoc;
data _null_;
dcl odsout trt();
trt.layout_absolute();
  2 trt.region(y: "2in", style_attr:"backgroundcolor=red");
    trt.format_text(data: "Orion Star", just: "c",
      style_attr: "color=cxbbb2e0 just=center font_size=72pt");
  3 trt.region(y: "3in", x: "3in", style_attr:"backgroundcolor=yellow");
    trt.format_text(data: "Sports & Outdoors",
      style_attr: "color=cxbbb2e0 font_size=28pt");
  4 trt.region(y: "5in", style_attr:"backgroundcolor=orange");
    trt.format_text(data: "Executive Prospectus",
      just: "c",
      style_attr: "font_size=36pt color=cxbbb2e0");
  5 trt.region(y: "7in", style_attr:"backgroundcolor=blue");
    trt.format_text(data: "For years 1999 through 2002",
      just: "c",
      style_attr: "font_size=20pt color=cxbbb2e0");
trt.layout_end();
run;
ods pdf close;
Program 4

Remove the region colors for your final output.

```sas
ods html close;
opts nodate nonumber;
title;
ods escapechar="-";
footnote "-{style [font_size=10pt just=right color=cxbbb2e0]
Provided to you by SAS 9.4 and ODS Absolute Layout features.}";
proc template;
  define style Styles.Orionbackground;
  parent=Styles.Printer;
  style body /
```
Chapter 11 • Reference and Examples

background=cx494068;
end;
run;
ods pdf file="your-file-path/Complete.pdf" style=Styles.Orionbackground notoc;
data _null_;
dcl odsout trt();
trt.layout_absolute();
  trt.region(y: "2in");
    trt.format_text(data: "Orion Star", just: "c",
                 style_attr: "color=cxbbb2e0 just=center font_size=72pt");
  trt.region(y: "3in", x: "3in");
    trt.format_text(data: "Sports & Outdoors",
                 style_attr: "color=cxbbb2e0 font_size=28pt");
  trt.region(y: "5in");
    trt.format_text(data: "Executive Prospectus",
                 just: "c",
                 style_attr: "font_size=36pt color=cxbbb2e0");
  trt.region(y: "7in");
    trt.format_text(data: "For years 1999 through 2002",
                 just: "c",
                 style_attr: "font_size=20pt color=cxbbb2e0");
trt.layout_end();
run;
ods pdf close;
Example 12: Creating a Cover Page with the LAYOUT_ABSOLUTE Method

Features:
- DECLARE statement
- Absolute Layout Methods
  - LAYOUT_ABSOLUTE
  - LAYOUT_END
  - REGION
- Text Method
  - FORMAT_TEXT

Output 4

Output 11.21  Final Output
Other features:
FOOTNOTE statement
ODS ESCAPECHAR statement
ODS PDF statement
TEMPLATE procedure

Details
The following example uses absolute layout methods to create a cover letter. Absolute layouts are useful to create one-page output. The output creates one layout container that contains four regions.

Program
ods html close;
options nodate nonumber;
title;
ods escapechar="~";
footnote "~{style \[font_size=10pt just=right color=cxbbb2e0\]Provided to you by SAS 9.4 and ODS Absolute Layout features.}";

proc template;
   define style Styles.Orionbackground;
   parent=Styles.Printer;
   style body /
      background=cx494068;
   end;
run;
ods pdf file="your-file-path/ExecutiveSummary.pdf" style=Styles.Orionbackground notoc;
data _null_; dcl odsout trt();
trt.layout_absolute();

   trt.region(y: "2in");
      trt.format_text(data: "Orion Star", just: "c",
          style_attr: "color=cxbbb2e0 just=center font_size=72pt");
   trt.region(y: "3in", x: "3in");
      trt.format_text(data: "Sports & Outdoors",
          style_attr:"color=cxbbb2e0 font_size=28pt");
   trt.region(y: "5in");
      trt.format_text(data: "Executive Prospectus",
          just: "c",
          style_attr:"font_size=36pt color=cxbbb2e0");
   trt.region(y: "7in");
      trt.format_text(data: "For years 1999 through 2002",
          just: "c",
          style_attr:"font_size=20pt color=cxbbb2e0");

trt.layout_end();
run;
Program Description

Close the HTML destination and set the SAS system options. The HTML destination is open by default. Absolute layouts are valid only in PRINTER destinations. Closing the HTML destination saves system resources.

ods html close;
options nodate nonumber;
title;

Define a representative character to be used with inline styles, and specify a footnote.

ods escapechar="~";
footnote "~[style {font_size=10pt just=right color=cxbbb2e0}Provided to you by SAS 9.4 and ODS Absolute Layout features.]

Create a style template for the output and open the PDF destination. The PROC TEMPLATE step creates the style Styles.Orionbackground to apply to the entire output.

proc template;
  define style Styles.Orionbackground;
    parent=Styles.Printer;
    style body /
      background=cx494068;
    end;
run;
ods pdf file="your-file-path/ExecutiveSummary.pdf" style=Styles.Orionbackground notoc;

Begin the DATA Null step and declare the local variable trt. The DATA _NULL_ statement begins the DATA step. The DECLARE statement assigns trt as the variable that identifies the object. The DECLARE statement also creates an instance.

data _null_; 
dcl odsout trt();

Begin the absolute layout. The LAYOUT_ABSOLUTE method begins an absolute layout container.

  trt.layout_absolute();

Create the body of the cover page. Each REGION method creates a region within the layout container. The X argument specifies the horizontal start position for each region. The Y argument specifies the vertical start position of the layout. The FORMAT_TEXT arguments display and format the text inside each region.

  trt.region(y: "2in");
    trt.format_text(data: "Orion Star", just: "c",
                   style_attr: "color=cxbbb2e0 just=center font_size=72pt");
  trt.region(y: "3in", x: "3in");
    trt.format_text(data: "Sports & Outdoors",
                    style_attr: "color=cxbbb2e0 font_size=28pt");
trt.region(y: "5in");
trt.format_text(data: "Executive Prospectus",
    just: "c",
    style_attr:"font_size=36pt color=cxbbb2e0");

trt.region(y: "7in");
trt.format_text(data: "For years 1999 through 2002",
    just: "c",
    style_attr:"font_size=20pt color=cxbbb2e0");

**End the layout.** The LAYOUT_END method ends the layout. The ODS PDF CLOSE statement closes the PDF destination.

    trt.layout_end();
    run;

    ods pdf close;
Example 12: Creating a Cover Page with the LAYOUT_ABSOLUTE Method

Output 11.22  Cover Page Created with the LAYOUT_ABSOLUTE Method

Orion Star
Sports & Outdoors

Executive Prospectus

For years 1999 through 2002
Chapter 12
Concepts

RWI Basics .......................................................... 249
DATA Step Processing ............................................ 249
Object-Oriented Concepts ......................................... 250
  Overview .............................................................. 250
  DATA Step and the Report Writing Interface ............... 250
  Using BY Groups ................................................. 254

RWI Basics

The RWI enables you to create and manipulate predefined ODS objects in a DATA step. ODS objects are data elements that consist of attributes, methods, and operators. Attributes are the properties that specify the information that is associated with an object. Methods define the operations that an object can perform. You use the DATA step object dot syntax to access the component object's attributes and methods. You can use the RWI’s object-oriented functionality to create highly specialized reports. The RWI is available for PRINTER and HTML destinations.

DATA Step Processing

DATA step processing has two distinct phases: the compilation phase and the execution phase. The compilation phase checks for syntax of the SAS statements and then compiles them, creating the input buffer, program data vector, and descriptor information.

The execution phase is executed once for each input observation that is being processed. The execution phase is further broken down into three categories: the initialization, the observation loop, and the termination processing. These three phases are particularly important to the Report Writing Interface.

When writing a custom report, you often have a heading section at the top, a summary section at the end, and probably a data-centric section in the middle. You can organize your ODS report writing code into each of these respective sections so that they are conditionally executed.
Object-Oriented Concepts

Overview

Objects are data structures combined with the associated processing routines. The new ODS object-oriented features in DATA _NULL_ report writing use the object-oriented programming technique. The basic concepts of object-oriented programming are object, class, instance, and method.

The following DATA step is the simplest program that you can run with the RWI.

```
data _null_;  
declare odsout obj();  
obj.format_text(data: 'Input data added with the FORMAT_TEXT method and the DATA argument.');  
... more methods ...  
run;  
```

1 Instantiation takes place with the DECLARE statement. An instance is the actual object created at run time. To create an instance, use the NEW statement with the DECLARE statement, or the DECLARE statement by itself.

2 ODSOUT is the class. ODSOUT is declared by the DECLARE statement. A class is a blueprint or template that describes how the output should look.

3 obj is the local variable. It is the name that identifies the object.

4 FORMAT_TEXT is a method. A method is the object’s application programming interface. A method can be called only by its own object. To call a method, use the object name followed by a dot. For example:

```
obj.format_text(data: 'Input Data added with the FORMAT_TEXT Method and the DATA argument.');  
```

5 DATA is an argument. It specifies input data for a method.

DATA Step and the Report Writing Interface

By default, when you specify a method with the RWI, the method or methods execute for every observation in the data set. In the following example, the FORMAT_TEXT method writes the text specified by the DATA argument for each observation in the Class data set. The variable object obj is also initialized once for each observation.

```
data _null_;  
set sashelp.class;  
declare odsout obj();  
obj.format_text(data: 'Text added with the FORMAT_TEXT method and the DATA argument.');  
run;  
```
You can specify multiple methods. Both methods are called for each observation.

data _null_
  set sashelp.class;
  declare odsout obj();
  obj.format_text(data: 'Text added with the FORMAT_TEXT');
  obj.format_text(data: 'method and the DATA argument.');
run;
Displaying the Value of a Variable Using the FORMAT(TEXT) Method

If you specify a variable name from the Class data set, the FORMAT(TEXT) method writes out the name of the variable for each observation. In the following example, Name and Sex are variables from Sashelp.Class.

```sas
data _null_
set sashelp.class;
declare odsout obj();
obj.format_text(data: name, data: sex);
run;
```
You can use the execution phase to organize your output. The object variable obj should be initialized only once for the duration of the DATA step (initialization execution phase). Therefore, the variable obj should be instantiated on the first observation of the observation execution loop. You can also specify any methods that you want to execute only on the first observation. Use IF-THEN processing to write out the text where you want it. In the following example, on the first observation, the DECLARE statement and the FORMAT_TEXT method that specifies a text string are executed. The FORMAT_TEXT method that specifies a variable name is then executed for each observation.

```sas
data _null_; set sashelp.class;
if _N_ = 1 then do;
    declare odsout obj();
    obj.format_text(data: 'The following names and weights are pulled in from the Class data set:');
    end;

    obj.format_text(data: name, data: weight);
run;
```
Using BY Groups

BY-group processing allows the program to take advantage of additional data-driven built-in logic. The DATA step identifies the beginning and ending of each BY group by creating additional temporary variables: FIRST.<variable> and LAST.<variable>. During the observation loop processing, on the first observation of a BY group, the value of FIRST.<variable> is set to 1. During the observation loop processing, on the last observation of a BY group, LAST.<variable> is set to 1.

```sas
proc sort data=sashelp.class out=class;
  by sex;
run;
data _null_; 
  set work.class;
  by sex;
  if _N_ = 1 then do;
    declare odsout obj();
  end;
  if first.sex then do;
    obj.format_text(data: sex);
  end;
  obj.format_text(data: name);
  if last.sex then do;
    obj.format_text(data: '');
  end;
```

Output 12.4  Using IF-THEN Processing to Control the Execution of Methods

The following names and weights are pulled in from the Class data set:
Alfred 112.5
Alice 84
Barbara 98
Carol 102.5
Henry 102.5
James 83
Jane 84.5
Janet 112.5
Jeffrey 84
John 99.5
Joyce 50.5
Judy 90
Louise 77
Mary 112
Philip 150
Robert 128
Ronald 133
Thomas 85
William 112
run;

**Output 12.5**  Using By Groups

<table>
<thead>
<tr>
<th>Using By-Groups</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
</tr>
<tr>
<td>Alice</td>
</tr>
<tr>
<td>Barbara</td>
</tr>
<tr>
<td>Carol</td>
</tr>
<tr>
<td>Jane</td>
</tr>
<tr>
<td>Janet</td>
</tr>
<tr>
<td>Joyce</td>
</tr>
<tr>
<td>Judy</td>
</tr>
<tr>
<td>Louise</td>
</tr>
<tr>
<td>Mary</td>
</tr>
<tr>
<td>M</td>
</tr>
<tr>
<td>Alfred</td>
</tr>
<tr>
<td>Henry</td>
</tr>
<tr>
<td>James</td>
</tr>
<tr>
<td>Jeffrey</td>
</tr>
<tr>
<td>John</td>
</tr>
<tr>
<td>Philip</td>
</tr>
<tr>
<td>Robert</td>
</tr>
<tr>
<td>Ronald</td>
</tr>
<tr>
<td>Thomas</td>
</tr>
<tr>
<td>William</td>
</tr>
</tbody>
</table>
Part 4

ODS Styles Reference

Chapter 13
  Overview ................................................................. 259

Chapter 14
  Style Templates ..................................................... 269

Chapter 15
  Style Elements ..................................................... 311

Chapter 16
  Style Attributes .................................................. 339
Understanding Styles, Style Elements, and Style Attributes

The appearance of SAS output is controlled by style templates (styles). A style is a type of ODS template that defines the visual aspects (colors, fonts, lines, markers, and so on) of SAS output. A style determines the overall look of the documents that use it. Style templates consist of style elements and style attributes.

- A style element is a named collection of style attributes that apply to a particular part of the output. Each area of ODS output has a style element name that is associated with it. The style element name specifies where the style attributes are applied. For example, a style element might contain instructions for the presentation of column headings or for the presentation of the data inside the cells. Style elements might also specify default colors and fonts for output that uses the style.

- A style attribute is a visual property, such as color, font properties, and line characteristics, that is defined in ODS with a reserved name and value. Style attributes are collectively referenced by a style element within a style template. Each style attribute specifies a value for one aspect of the presentation. For example, the BACKGROUNDCOLOR= attribute specifies the color for the background of an HTML table or for a colored table in printed output. The FONTSTYLE= attribute specifies whether to use a Roman font or an italic font.

Note: Because styles control the presentation of the data, they have no effect on output objects that go to the LISTING, DOCUMENT, or OUTPUT destination.

Available styles are in the SASHELP.TMPLMST item store. In SAS Enterprise Guide, the list of style sheets is shown by the Style Wizard. In batch mode or SAS Studio, you can display the list of available style templates by submitting this code:

```sas
proc template;
list styles / store=sashelp.tmplmst;
run;
```

For complete information about viewing ODS styles, see “Viewing ODS Styles Supplied by SAS” on page 269.
By default, HTML output uses the HTMLBlue style template. To help you become familiar with styles, style elements, and style attributes, look at the relationship between them. The diagram that follows shows the relationship between the style, the style elements, and the style attributes. The following figure illustrates the structure of a style:
Template Browser

proc template;
    define style Styles.HTMLBlue;
        parent = styles.statistical;
        class GraphColors /
            'gblockheader' = cxcfd5de
            'gphasebox' = cx9b9Ea1
            'gphasebox' = cxDBe6F2
            'gzonec' = cxEbCee0
            'gzonec' = cxCcDcee
            'gzoneb' = cxCcDcee
            'gzoneb' = cxD7E5F3
            'gzonea' = cxE3Edf7
            'gconramp3cend' = cx9C1C00
            'gconramp3cneutral' = cx222222
            'gconramp3cstart' = cxe0E36Ac
            'gramp3cend' = cxD05B5B
            'gramp3cneutral' = cxfAFBFE
            'gramp3cstart' = cxe667FA2
            'gcontrollim' = cxEBF2FF
            'gcontrolim' = cxBFCTD9
            'gruntest' = cxCAE3FF
            'gcruntest' = cxBF4D4D
            'gclipping' = cxFFFFC6
            'gclipping' = cxC1C100
    ...
    class Header /
        bordercolor = cxB087BB
        backgroundcolor = cxDfF2F9
        color = cx112277;
    class Footer /
        bordercolor = cxB087BB
        backgroundcolor = cxDfF2F9
        color = cx112277;
    class RowHeader /
        bordercolor = cxB087BB
        backgroundcolor = cxDfF2F9
        color = cx112277;
    class RowFooter /
        bordercolor = cxB087BB
        backgroundcolor = cxDfF2F9
        color = cx112277;
    class Table /
        cellpadding = 5;
    class Graph /
    attrpriority = "Color";
    class GraphFit2 /
        linestyle = 1;
    class GraphClipping /
        markersymbol = "circlefilled";
end;
run;
*** END OF TEXT ***
The following list corresponds to the numbered items in the preceding figure:

1. Styles.HtmlBlue is the style. Styles describe how to display presentation aspects (color, font, font size, and so on) of the SAS output. A style determines the overall appearance of the ODS documents that use it. The default style for HTML output is HtmlBlue. Each style consists of style elements. Each destination has a default style that is applied to all output that is written to the destination.
   - The default style for HTML output is HTMLBlue.
   - The default style for PRINTER output is Pearl.
   - The default style for RTF output is RTF.

You can create new styles with the “DEFINE STYLE Statement” in SAS Output Delivery System: Procedures Guide. New styles can be created independently or from an existing style. You can use “PARENT= Statement” in SAS Output Delivery System: Procedures Guide to create a new style from an existing style. For complete documentation about ODS styles, see “Style Templates” in SAS Output Delivery System: User's Guide.

2. Header and Footer are examples of style elements. A style element is a collection of style attributes that apply to a particular part of the output for a SAS program. For example, a style element might contain instructions for the presentation of column headings or for the presentation of the data inside table cells. Style elements might also specify default colors and fonts for output that uses the style. Style elements exist inside styles and consist of one or more style attributes. Style elements can be user-defined or supplied by SAS. User-defined style elements can be created by the “STYLE Statement” in SAS Output Delivery System: Procedures Guide.

   Note: For a list of the default style elements used for HTML and markup languages and their inheritance, see “Style Elements” in SAS Output Delivery System: User's Guide.

3. BORDERCOLOR=, BACKGROUNDCOLOR=, and COLOR= are examples of style attributes. Style attributes specify a value for one aspect of the area of the output that its style element applies to. For example, the COLOR= attribute specifies the value cx112277 for the font color. For a list of style attributes supplied by SAS, see “Style Attributes” in SAS Output Delivery System: User's Guide.

   Style attributes can be referenced with style references. See “style-reference” on page 387 for more information about style references.

The following table shows commonly used style attributes that you can set with the STYLE= option in PROC PRINT, PROC TABULATE, and PROC REPORT. Most of these attributes apply to parts of the table other than cells (for example, table borders and the lines between columns and rows). Note that not all attributes are valid in all destinations. For more information about these style attributes, their valid values, and their applicable destinations, see “Style Attributes Tables” in SAS Output Delivery System: Procedures Guide.
### Table 13.1  Style Attributes for PROC REPORT, PROC TABULATE, and PROC PRINT

<table>
<thead>
<tr>
<th>Attribute</th>
<th>PROC REPORT STATEMENT REPORT Area</th>
<th>PROC REPORT Areas: CALLDEF, COLUMN, HEADER, LINES, SUMMARY</th>
<th>PROC TABULATE STATEMENT TABLE</th>
<th>PROC TABULATE STATEMENTS VAR, CLASS, BOX, CLASSLEV, KEYWORD</th>
<th>PROC PRINT TABLE location</th>
<th>PROC PRINT: all locations other than TABLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASIS=</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>BACKGROUNDCOLOR =</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>BACKGROUNDMAGE =</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>BORDERBOTTOMCOLOR =</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>BORDERBOTTOMSTYLE =</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>BORDERBOTTOMWIDTH =</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>BORDERLEFTCOLOR =</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>BORDERLEFTSTYLE =</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>BORDERLEFTWIDTH =</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>BORDERCOLOR =</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>BORDERCOLORDARK =</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>BORDERCOLORLIGHT =</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>BORDERRIGHTCOLOR =</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>BORDERRIGHTSTYLE =</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>BORDERRIGHTWIDTH =</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>BORDERTOPCOLOR =</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>BORDERTOPSTYLE =</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>BORDERTOPWIDTH =</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>BORDERWIDTH =</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Attribute</td>
<td>PROC REPORT STATEMENT REPORT Area</td>
<td>PROC REPORT Areas: CALLDEF, COLUMN, HEADER, LINES, SUMMARY</td>
<td>PROC TABULATE STATEMENT TABLE</td>
<td>PROC TABULATE STATEMENTS VAR, CLASS, BOX, CLASSLEV, KEYWORD</td>
<td>PROC PRINT: all locations other than TABLE</td>
<td></td>
</tr>
<tr>
<td>--------------------</td>
<td>----------------------------------</td>
<td>-----------------------------------------------------------</td>
<td>-------------------------------</td>
<td>------------------------------------------------------------</td>
<td>-------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>CELLPADDING=</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CELLPACING=</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CELLWIDTH=</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>CLASS=</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>COLOR=</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FLYOVER=</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>FONT=</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>FONTFAMILY=</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>FONTSIZE=</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>FONTSTYLE=</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>FONTWEIGHT=</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>FONTWIDTH=</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FRAME=</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>HEIGHT=</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>HREFTARGET=</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>HTMLSTYLE=</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NOBREAKSPACE=</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>OUTPUTWIDTH=</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>POSTHTML=’*</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>POSTIMAGE=</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>POSTTEXT=’*</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>PREHTML=’*</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>PREIMAGE=</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>PRETEXT=’*</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>PROTECTSPECIALCHARS=</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
</tbody>
</table>
Using Styles with Base SAS Procedures

Most Base SAS procedures that support ODS use one or more table templates to produce output objects. These table templates include templates for table elements: columns, headers, and footers. Each table element can specify the use of one or more style elements for various parts of the output. These style elements cannot be specified within the syntax of the procedure, but you can use customized styles for the ODS destinations that you use. For more information about customizing tables and styles, see “TEMPLATE Procedure: Creating a Style Template” in SAS Output Delivery System: Procedures Guide.

The Base SAS reporting procedures, PROC PRINT, PROC REPORT, and PROC TABULATE, enable you to quickly analyze your data and organize it into easy-to-read tables. You can use the STYLE= option with these procedure statements to modify the appearance of your report. The STYLE= option enables you to make changes in sections of output without changing the default style for all of the output. You can customize specific sections of procedure output by specifying the STYLE= option in specific statements within the procedure.

The following program uses the STYLE= option to create the background colors in the PROC REPORT output below:

```sas
title "Height and Weight by Gender and Age";
proc report nowd data=sashelp.class
    style(header)= [background=white];
    col age ('Gender' sex),(weight height));
    define age / style(header)= [background=lightgreen];
    define sex / across style(header)= [background=yellow] ' ';
    define weight / style(header)= [background=orange];
    define height / style(header)= [background=tan];
```

* When you use these attributes in this location, they affect only the text that is specified with the PRETEXT=, POSTTEXT=, PREHTML=, and POSTHTML= attributes. To alter the foreground color or the font for the text that appears in the table, you must set the corresponding attribute in a location that affects the cells rather than the table. For complete documentation about style attributes and their values, see Chapter 16, “Style Attributes,” on page 339.
Figure 13.2 Enhanced PROC REPORT Output

The following program uses the STYLE= option to create the colors in the PROC TABULATE output below:

```sas
proc sort data=sashelp.prdsale out=prdsale;
   by Country;
run;

proc tabulate data=prdsale;
   class region division prodtype / style=[background=lightgreen];
   classlev region division prodtype / style=[background=yellow];
   var actual / style=[background=tan];
   keyword all sum / style=[background=linen color=blue];
   keylabel all='Total';
   table (region all)*(division all),
      (prodtype all)*(actual*f=dollar10.) /
      box=[label='Region by Division and Type' style=[backgroundcolor=orange]];

   title 'Actual Product Sales';
   title2 '(millions of dollars)';
run;
```
The following program uses the STYLE= option to create the colors in the PROC PRINT output below:

```sas
proc print data=exprev noobs sumlabel='Total' GRANDTOTAL_LABEL="Grand Total"
  style(table)=\{frame=box rules=groups\}
  style(bysumline)=\{background=red foreground=linen\}
  style(grandtotal)=\{foreground=green\}
  style(header)=\{font_style=italic background=orange\};
by sale_type order_date;
sum price quantity;
sumby sale_type;
label sale_type='Sale Type' order_date='Sale Date';
format price dollar10.2 cost dollar10.2;
var Country / style(data)=\{font_face=arial font_weight=bold background=linen\};
var Price / style(data)=\{font_style=italic background=yellow\};
var Cost / style(data)=\{foreground=hgt. background=lightgreen\};
title 'Retail and Quantity Totals for Each Sale Type';
run;
```

For the complete input data set, see “EXPREV” in *Base SAS Procedures Guide*. 
### Retail and Quantity Totals for Each Sale Type

#### Sale Type=Catalog Sale Date=1/1/12

<table>
<thead>
<tr>
<th>Country</th>
<th>Price</th>
<th>Cost</th>
<th>Quantity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Puerto Rico</td>
<td>$51.20</td>
<td>$12.10</td>
<td>14</td>
</tr>
<tr>
<td>Aruba</td>
<td>$123.70</td>
<td>$59.00</td>
<td>30</td>
</tr>
<tr>
<td>Bahamas</td>
<td>$113.40</td>
<td>$28.45</td>
<td>8</td>
</tr>
<tr>
<td>Bermuda</td>
<td>$41.00</td>
<td>$9.25</td>
<td>7</td>
</tr>
</tbody>
</table>

#### Sale Type=Catalog Sale Date=1/2/12

<table>
<thead>
<tr>
<th>Country</th>
<th>Price</th>
<th>Cost</th>
<th>Quantity</th>
</tr>
</thead>
<tbody>
<tr>
<td>British Virgin Islands</td>
<td>$40.20</td>
<td>$20.20</td>
<td>11</td>
</tr>
<tr>
<td>Canada</td>
<td>$11.80</td>
<td>$5.00</td>
<td>100</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>$381.30</strong></td>
<td><strong>170</strong></td>
<td></td>
</tr>
</tbody>
</table>

#### Sale Type=In Store Sale Date=1/1/12

<table>
<thead>
<tr>
<th>Country</th>
<th>Price</th>
<th>Cost</th>
<th>Quantity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Virgin Islands (U.S.)</td>
<td>$31.10</td>
<td>$15.66</td>
<td>25</td>
</tr>
</tbody>
</table>

#### Sale Type=In Store Sale Date=1/2/12

<table>
<thead>
<tr>
<th>Country</th>
<th>Price</th>
<th>Cost</th>
<th>Quantity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Belize</td>
<td>$146.40</td>
<td>$36.70</td>
<td>2</td>
</tr>
<tr>
<td>Cayman Islands</td>
<td>$71.00</td>
<td>$32.30</td>
<td>20</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>$248.50</strong></td>
<td><strong>47</strong></td>
<td></td>
</tr>
</tbody>
</table>

#### Sale Type=Internet Sale Date=1/1/12

<table>
<thead>
<tr>
<th>Country</th>
<th>Price</th>
<th>Cost</th>
<th>Quantity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Antarctica</td>
<td>$92.60</td>
<td>$20.70</td>
<td>2</td>
</tr>
<tr>
<td><strong>Grand Total</strong></td>
<td><strong>$722.40</strong></td>
<td></td>
<td><strong>219</strong></td>
</tr>
</tbody>
</table>
Chapter 14
Style Templates

Viewing ODS Styles Supplied by SAS

To view the names of all of the style templates that are shipped with SAS, submit the following program. The style templates are all located in the Styles folder by default.

```sas
proc template;
   list styles;
run;
```

To view the source code of a specified style, submit the following code:

```sas
proc template;
   source styles.style-name;
run;
```
Figure 14.1 Styles Supplied by SAS

Log 14.1 Partial Source Code for the HTMLBlue Style

```sas
9    proc template;
10      source styles.htmlBlue;
define style Styles.htmlBlue;
      parent = styles.statistical;
      class GraphColors /
         'gndata12' = cxECE8C4
         'gndata11' = cxDBD8F8
         'gndata10' = cxC6E4BF
         'gndata9' = cxE6CEAD
         'gndata8' = cxE5C1D4
         'gndata7' = cxC9DFF0
         'gndata6' = cxDD8EB5
         'gndata5' = cxDBC7E7
         'gndata4' = cxD5C6B4
         'gndata3' = cxB7D4D3
         'gndata2' = cxE7B3B4
         'gndata1' = cxBBC2DC
         'gndata' = cxC8C9CB
```

Note: If you are using SAS Studio, you do not need to specify the STYLE= option. You can go to Preferences ➤ Results and change the style from the drop-down list for your selected destination.
Table of Suggested ODS Styles

With ODS, you can use any style with any output destination. However, for each destination, SAS supplies one or more styles that are optimized to work with the output the destination creates.

**Table 14.1  Recommended Styles for ODS Destinations**

<table>
<thead>
<tr>
<th>Destination</th>
<th>Recommended Styles</th>
<th>Default Style</th>
</tr>
</thead>
<tbody>
<tr>
<td>EPUB</td>
<td>Daisy</td>
<td>Daisy</td>
</tr>
<tr>
<td></td>
<td>Moonflower</td>
<td></td>
</tr>
<tr>
<td>ODS destination for Excel</td>
<td>Excel</td>
<td>Excel</td>
</tr>
<tr>
<td>Printer family of statements</td>
<td>FancyPrinter</td>
<td>Pearl</td>
</tr>
<tr>
<td></td>
<td>FestivalPrinter</td>
<td></td>
</tr>
<tr>
<td></td>
<td>GrayscalePrinter</td>
<td></td>
</tr>
<tr>
<td></td>
<td>MeadowPrinter</td>
<td></td>
</tr>
<tr>
<td></td>
<td>MonoChromePrinter</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Monospace</td>
<td></td>
</tr>
<tr>
<td></td>
<td>NormalPrinter</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Pearl</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Printer</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Sapphire</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SasDocPrinter</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SeasidePrinter</td>
<td></td>
</tr>
<tr>
<td>RTF</td>
<td>RTF</td>
<td>RTF</td>
</tr>
<tr>
<td>TAGSETS.RTF</td>
<td>RTF</td>
<td>RTF</td>
</tr>
<tr>
<td>ODS destination for PowerPoint</td>
<td>PowerPointDark</td>
<td>PowerPointLight</td>
</tr>
<tr>
<td></td>
<td>PowerPointLight</td>
<td></td>
</tr>
<tr>
<td>LISTING</td>
<td>Listing</td>
<td>Listing</td>
</tr>
</tbody>
</table>
Program for Viewing Multiple Styles

This program creates a sample report in HTML, PDF, and RTF of every style supplied by SAS. The output appears in your working directory. Although you can apply most

---

* The Moonflower style for ODS EPUB is designed for nighttime or low-light reading.
Program for Viewing Multiple Styles

273

SAS styles to any destination, SAS supplies one or more styles that are optimized to
work with the output the destination creates. For a table of suggested ODS styles for
each destination, see Table 14.1 on page 271.
The table Gallery is created and is used in a subsequent DATA step to generate a list of
each style for each destination. The destination link is a hyperlink.
ods _all_ close;
proc template;
define table gallery;
column libname memname style
define libname ;
blank_dups=on;
end;
define links;
header = 'Samples';
compute as '<a href="' ||
'<a href="' ||
'<a href="' ||
end;
end;
run;

links;

trim(style) ||'.html">HTML</a> ' ||
trim(style) ||'.pdf">PDF</a> ' ||
trim(style) ||'.rtf">RTF</a>';

The DATA step creates an index of all available styles supplied by SAS from the
template store Sashelp.Tmplmst.
ods html file="index.html";
title "Index of all styles";
data _null_;
set sashelp.Gallery(where=(libname="SASHELP"));
file print ods=(template='Gallery');
put _ods_;
run;
ods html close;

The ODS destination statements create the output. You can add additional destinations
by specifying the following statement for each destination: ods destination
file="&style..destination-extension" style=&style;
%macro generateods();
options nodate;
ods html file="&style..html" style=&style;
ods pdf file="&style..pdf" style=&style;
ods rtf file="&style..rtf" style=&style;
title "Style is: &style";

The ODS NOPTITLE statement removes the procedure title.
ods noptitle;

The ODS SELECT statement selects the Variables table for the gallery.
ods select variables;
proc contents data=sashelp.class;
run;
ods _all_ close;
%mend;


The ODS NORESULTS statement prevents an entry in the results window for each of the subsequent PROC CONTENTS steps that are generated.

```sas
ods noresults;
```

This DATA step creates a sample of each style.

```sas
data _null_;  
    set sashelp.Gallery(where=(libname="SASHELP"));  
    call symputx('style', style);  
    call execute('%generateods');  
run;
```

The ODS RESULTS and ODS PREFERENCES statements set the ODS options back to defaults.

```sas
ods results;  
ods preferences;
```

---

**ODS Styles Gallery**

**EPUB Daisy Style**

*Output 14.1  EPUB Book Title Page*
**HTML Styles**

You can view and modify the default HTML style by selecting **Tools \(\Rightarrow\) Options \(\Rightarrow\) Preferences** from the menu at the top of the main SAS window. Then open the **Results** tab. You can change the style by selecting a style from the **Style** drop-down menu. The settings in your Preferences window persist until you explicitly change them. The following display shows the **Results** tab with the new HTML style specified:

**Note:** If you are using SAS Studio, you do not need to specify the **STYLE=** option. You can go to **Preferences \(\Rightarrow\) Results** and change the style from the drop-down list for your selected destination.
Figure 14.2  Changing the HTML Style with the Preferences Window
Figure 14.3 Changing the HTML Style with SAS Studio
Output 14.5  Analysis Style
Output 14.6  BarrettsBlue Style

---

Two bar charts showing sales for different regions and products (BED, CHAIR, DESK, SOFA, TABLE) with predicted vs. actual sales indicated.

---

A scatter plot showing weight vs. height with confidence and prediction limits and a regression line.

---

A table with data for sofa sales:

<table>
<thead>
<tr>
<th>Obs</th>
<th>PRODUCT</th>
<th>REGION</th>
<th>ACTUAL</th>
<th>PREDICT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>SOFA</td>
<td>EAST</td>
<td>$925.00</td>
<td>$850.00</td>
</tr>
<tr>
<td>2</td>
<td>SOFA</td>
<td>EAST</td>
<td>$999.00</td>
<td>$297.00</td>
</tr>
<tr>
<td>3</td>
<td>SOFA</td>
<td>EAST</td>
<td>$608.00</td>
<td>$646.00</td>
</tr>
<tr>
<td>4</td>
<td>SOFA</td>
<td>EAST</td>
<td>$642.00</td>
<td>$533.00</td>
</tr>
<tr>
<td>5</td>
<td>SOFA</td>
<td>EAST</td>
<td>$656.00</td>
<td>$646.00</td>
</tr>
<tr>
<td>6</td>
<td>SOFA</td>
<td>EAST</td>
<td>$948.00</td>
<td>$456.00</td>
</tr>
<tr>
<td>7</td>
<td>SOFA</td>
<td>EAST</td>
<td>$612.00</td>
<td>$717.00</td>
</tr>
<tr>
<td>8</td>
<td>SOFA</td>
<td>EAST</td>
<td>$114.00</td>
<td>$584.00</td>
</tr>
</tbody>
</table>
Output 14.7  BlockPrint Style

![Bar Chart](chart1.png)

![Line Chart](chart2.png)

<table>
<thead>
<tr>
<th>Obs</th>
<th>PRODUCT</th>
<th>REGION</th>
<th>ACTUAL</th>
<th>PREDICT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>SOFA</td>
<td>EAST</td>
<td>$923.00</td>
<td>$950.00</td>
</tr>
<tr>
<td>2</td>
<td>SOFA</td>
<td>EAST</td>
<td>$999.00</td>
<td>$977.00</td>
</tr>
<tr>
<td>3</td>
<td>SOFA</td>
<td>EAST</td>
<td>$608.00</td>
<td>$646.00</td>
</tr>
<tr>
<td>4</td>
<td>SOFA</td>
<td>EAST</td>
<td>$642.00</td>
<td>$553.00</td>
</tr>
<tr>
<td>5</td>
<td>SOFA</td>
<td>EAST</td>
<td>$656.00</td>
<td>$646.00</td>
</tr>
<tr>
<td>6</td>
<td>SOFA</td>
<td>EAST</td>
<td>$948.00</td>
<td>$986.00</td>
</tr>
<tr>
<td>7</td>
<td>SOFA</td>
<td>EAST</td>
<td>$612.00</td>
<td>$717.00</td>
</tr>
</tbody>
</table>
Output 14.8  Default Style

![Graph showing sales data for different regions and products.](image)

<table>
<thead>
<tr>
<th>Obs</th>
<th>PRODUCT</th>
<th>REGION</th>
<th>ACTUAL</th>
<th>PREDICT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>SOFA</td>
<td>EAST</td>
<td>$925.00</td>
<td>$850.00</td>
</tr>
<tr>
<td>2</td>
<td>SOFA</td>
<td>EAST</td>
<td>$999.00</td>
<td>$997.00</td>
</tr>
<tr>
<td>3</td>
<td>SOFA</td>
<td>EAST</td>
<td>$608.00</td>
<td>$646.00</td>
</tr>
<tr>
<td>4</td>
<td>SOFA</td>
<td>EAST</td>
<td>$642.00</td>
<td>$533.00</td>
</tr>
<tr>
<td>5</td>
<td>SOFA</td>
<td>EAST</td>
<td>$656.00</td>
<td>$646.00</td>
</tr>
<tr>
<td>6</td>
<td>SOFA</td>
<td>EAST</td>
<td>$948.00</td>
<td>$486.00</td>
</tr>
<tr>
<td>7</td>
<td>SOFA</td>
<td>EAST</td>
<td>$612.00</td>
<td>$717.00</td>
</tr>
</tbody>
</table>
Output 14.9  Dove Style

![Graph showing sales and region for different furniture items, including beds, chairs, desks, sofas, and tables.](image)

![Graph showing a scatter plot with height and weight, with 95% confidence and prediction limits.](image)

![Table showing sales data for sofas in different regions.](table)

<table>
<thead>
<tr>
<th>Obs</th>
<th>PRODUCT</th>
<th>REGION</th>
<th>ACTUAL</th>
<th>PREDICT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>SOFA</td>
<td>EAST</td>
<td>$925.00</td>
<td>$850.00</td>
</tr>
<tr>
<td>2</td>
<td>SOFA</td>
<td>EAST</td>
<td>$999.00</td>
<td>$987.00</td>
</tr>
<tr>
<td>3</td>
<td>SOFA</td>
<td>EAST</td>
<td>$958.00</td>
<td>$846.00</td>
</tr>
<tr>
<td>4</td>
<td>SOFA</td>
<td>EAST</td>
<td>$942.00</td>
<td>$553.00</td>
</tr>
<tr>
<td>5</td>
<td>SOFA</td>
<td>EAST</td>
<td>$858.00</td>
<td>$646.00</td>
</tr>
<tr>
<td>6</td>
<td>SOFA</td>
<td>EAST</td>
<td>$946.00</td>
<td>$496.00</td>
</tr>
</tbody>
</table>
Output 14.10  Dtree Style

The diagram shows the predicted and actual sales for various products in different regions. The sales data is represented for 'BED', 'CHAIR', 'DESK', 'SOFA', and 'TABLE'. The x-axis represents the regions (EAST, WEST), and the y-axis represents the sales values. The chart includes a comparison of predicted vs. actual sales.

Below is a table summarizing the observed sales data:

<table>
<thead>
<tr>
<th>Obs</th>
<th>PRODUCT</th>
<th>REGION</th>
<th>ACTUAL</th>
<th>PREDICT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>SOFA</td>
<td>EAST</td>
<td>$925.00</td>
<td>$850.00</td>
</tr>
<tr>
<td>2</td>
<td>SOFA</td>
<td>EAST</td>
<td>$999.00</td>
<td>$297.00</td>
</tr>
<tr>
<td>3</td>
<td>SOFA</td>
<td>EAST</td>
<td>$608.00</td>
<td>$846.00</td>
</tr>
<tr>
<td>4</td>
<td>SOFA</td>
<td>EAST</td>
<td>$642.00</td>
<td>$533.00</td>
</tr>
<tr>
<td>5</td>
<td>SOFA</td>
<td>EAST</td>
<td>$656.00</td>
<td>$646.00</td>
</tr>
<tr>
<td>6</td>
<td>SOFA</td>
<td>EAST</td>
<td>$948.00</td>
<td>$486.00</td>
</tr>
<tr>
<td>7</td>
<td>SOFA</td>
<td>EAST</td>
<td>$612.00</td>
<td>$717.00</td>
</tr>
<tr>
<td>8</td>
<td>SOFA</td>
<td>EAST</td>
<td>$614.00</td>
<td>$564.00</td>
</tr>
</tbody>
</table>
Output 14.11  EGDefault Style

---

**Diagram 1:**
- **Graph Type:** Bar chart
- **X-axis:** Region (EAST, YEST)
- **Y-axis:** Sales
- **Legend:**
  - Predicted Sales
  - Actual Sales

**Diagram 2:**
- **Graph Type:** Scatter plot
- **X-axis:** Height
- **Y-axis:** Weight
- **Legend:**
  - 95% Confidence Limits
  - 95% Prediction Limits
  - Regression

---

**Table:**

<table>
<thead>
<tr>
<th>Obs</th>
<th>PRODUCT</th>
<th>REGION</th>
<th>ACTUAL</th>
<th>PREDICT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>SOFA</td>
<td>EAST</td>
<td>$925.00</td>
<td>$950.00</td>
</tr>
<tr>
<td>2</td>
<td>SOFA</td>
<td>EAST</td>
<td>$999.00</td>
<td>$997.00</td>
</tr>
<tr>
<td>3</td>
<td>SOFA</td>
<td>EAST</td>
<td>$608.00</td>
<td>$646.00</td>
</tr>
<tr>
<td>4</td>
<td>SOFA</td>
<td>EAST</td>
<td>$642.00</td>
<td>$633.00</td>
</tr>
<tr>
<td>5</td>
<td>SOFA</td>
<td>EAST</td>
<td>$656.00</td>
<td>$646.00</td>
</tr>
<tr>
<td>6</td>
<td>SOFA</td>
<td>EAST</td>
<td>$940.00</td>
<td>$936.00</td>
</tr>
<tr>
<td>7</td>
<td>SOFA</td>
<td>EAST</td>
<td>$612.00</td>
<td>$717.00</td>
</tr>
<tr>
<td>8</td>
<td>SOFA</td>
<td>EAST</td>
<td>$114.00</td>
<td>$564.00</td>
</tr>
</tbody>
</table>
Output 14.12  Festival Style

---

**Table:**

<table>
<thead>
<tr>
<th>Obs</th>
<th>PRODUCT</th>
<th>REGION</th>
<th>ACTUAL</th>
<th>PREDICT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>SOFA</td>
<td>EAST</td>
<td>$625.00</td>
<td>$600.00</td>
</tr>
<tr>
<td>2</td>
<td>SOFA</td>
<td>EAST</td>
<td>$686.00</td>
<td>$267.00</td>
</tr>
<tr>
<td>3</td>
<td>SOFA</td>
<td>EAST</td>
<td>$600.00</td>
<td>$646.00</td>
</tr>
<tr>
<td>4</td>
<td>SOFA</td>
<td>EAST</td>
<td>$642.00</td>
<td>$533.00</td>
</tr>
<tr>
<td>5</td>
<td>SOFA</td>
<td>EAST</td>
<td>$686.00</td>
<td>$646.00</td>
</tr>
<tr>
<td>6</td>
<td>SOFA</td>
<td>EAST</td>
<td>$648.00</td>
<td>$486.00</td>
</tr>
<tr>
<td>7</td>
<td>SOFA</td>
<td>EAST</td>
<td>$612.00</td>
<td>$717.00</td>
</tr>
<tr>
<td>8</td>
<td>SOFA</td>
<td>EAST</td>
<td>$114.00</td>
<td>$504.00</td>
</tr>
</tbody>
</table>
Output 14.13  Gantt Style

The Gantt Style output includes a series of bar charts showing sales data for different products (BED, CHAIR, DESK, SOFA, TABLE) across regions (EAST, WEST). The bars represent the predicted sales, while the lines represent the actual sales.

Additionally, there is a scatter plot showing the relationship between weight and height, with lines indicating 95% confidence limits and 95% prediction limits. The table below provides actual and predicted sales for different products in the EAST region:

<table>
<thead>
<tr>
<th>Obs</th>
<th>PRODUCT</th>
<th>REGION</th>
<th>ACTUAL</th>
<th>PREDICT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>SOFA</td>
<td>EAST</td>
<td>$925.00</td>
<td>$850.00</td>
</tr>
<tr>
<td>2</td>
<td>SOFA</td>
<td>EAST</td>
<td>$999.00</td>
<td>$297.00</td>
</tr>
<tr>
<td>3</td>
<td>SOFA</td>
<td>EAST</td>
<td>$608.00</td>
<td>$646.00</td>
</tr>
<tr>
<td>4</td>
<td>SOFA</td>
<td>EAST</td>
<td>$642.00</td>
<td>$533.00</td>
</tr>
<tr>
<td>5</td>
<td>SOFA</td>
<td>EAST</td>
<td>$686.00</td>
<td>$646.00</td>
</tr>
<tr>
<td>6</td>
<td>SOFA</td>
<td>EAST</td>
<td>$948.00</td>
<td>$486.00</td>
</tr>
<tr>
<td>7</td>
<td>SOFA</td>
<td>EAST</td>
<td>$612.00</td>
<td>$717.00</td>
</tr>
<tr>
<td>8</td>
<td>SOFA</td>
<td>EAST</td>
<td>$644.00</td>
<td>$686.00</td>
</tr>
</tbody>
</table>
Output 14.14  Harvest Style

![Graph showing sales and regions]

![Graph showing weight and height with confidence and prediction limits]

<table>
<thead>
<tr>
<th>Obs</th>
<th>PRODUCT</th>
<th>REGION</th>
<th>ACTUAL</th>
<th>PREDICT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>SOFA</td>
<td>EAST</td>
<td>$925.03</td>
<td>$900.03</td>
</tr>
<tr>
<td>2</td>
<td>SOFA</td>
<td>EAST</td>
<td>$900.03</td>
<td>$927.03</td>
</tr>
<tr>
<td>3</td>
<td>SOFA</td>
<td>EAST</td>
<td>$938.03</td>
<td>$949.03</td>
</tr>
<tr>
<td>4</td>
<td>SOFA</td>
<td>EAST</td>
<td>$942.03</td>
<td>$933.03</td>
</tr>
<tr>
<td>5</td>
<td>SOFA</td>
<td>EAST</td>
<td>$895.03</td>
<td>$848.03</td>
</tr>
<tr>
<td>6</td>
<td>SOFA</td>
<td>EAST</td>
<td>$946.03</td>
<td>$949.03</td>
</tr>
<tr>
<td>7</td>
<td>SOFA</td>
<td>EAST</td>
<td>$912.03</td>
<td>$717.03</td>
</tr>
<tr>
<td>8</td>
<td>SOFA</td>
<td>EAST</td>
<td>$914.03</td>
<td>$904.03</td>
</tr>
</tbody>
</table>
Output 14.15  HighContrast Style

Top: Bar charts showing sales for different regions (EAST and WEST) for various products (BED, CHAIR, DESK, SOFA, TABLE).

Middle: Scatter plot with 95% confidence and prediction limits, showing the relationship between height and weight.

Bottom: Table listing observations with product, region, actual, and predicted sales.
Output 14.16  HTMLBlue Style

![Bar chart showing sales data for different products across regions.]

![Scatter plot with regression line and 95% confidence limits.]

<table>
<thead>
<tr>
<th>Obs</th>
<th>PRODUCT</th>
<th>REGION</th>
<th>ACTUAL</th>
<th>PREDICT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>SOFA</td>
<td>EAST</td>
<td>$625.00</td>
<td>$550.00</td>
</tr>
<tr>
<td>2</td>
<td>SOFA</td>
<td>EAST</td>
<td>$998.00</td>
<td>$297.00</td>
</tr>
<tr>
<td>3</td>
<td>SOFA</td>
<td>EAST</td>
<td>$608.00</td>
<td>$546.00</td>
</tr>
<tr>
<td>4</td>
<td>SOFA</td>
<td>EAST</td>
<td>$462.00</td>
<td>$533.00</td>
</tr>
<tr>
<td>5</td>
<td>SOFA</td>
<td>EAST</td>
<td>$666.00</td>
<td>$546.00</td>
</tr>
<tr>
<td>6</td>
<td>SOFA</td>
<td>EAST</td>
<td>$946.00</td>
<td>$466.00</td>
</tr>
<tr>
<td>7</td>
<td>SOFA</td>
<td>EAST</td>
<td>$612.00</td>
<td>$717.00</td>
</tr>
<tr>
<td>8</td>
<td>SOFA</td>
<td>EAST</td>
<td>$714.00</td>
<td>$564.00</td>
</tr>
</tbody>
</table>
Output 14.17  Journal Style

![Graph showing sales comparison for different regions and products.](image)

![Graph showing weight and height correlation.](image)

<table>
<thead>
<tr>
<th>Obs</th>
<th>PRODUCT</th>
<th>REGION</th>
<th>ACTUAL</th>
<th>PREDICT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>SOFA</td>
<td>EAST</td>
<td>825.00</td>
<td>850.00</td>
</tr>
<tr>
<td>2</td>
<td>SOFA</td>
<td>EAST</td>
<td>999.00</td>
<td>927.00</td>
</tr>
<tr>
<td>3</td>
<td>SOFA</td>
<td>EAST</td>
<td>833.00</td>
<td>848.00</td>
</tr>
<tr>
<td>4</td>
<td>SOFA</td>
<td>EAST</td>
<td>842.00</td>
<td>853.00</td>
</tr>
<tr>
<td>5</td>
<td>SOFA</td>
<td>EAST</td>
<td>805.00</td>
<td>890.00</td>
</tr>
<tr>
<td>6</td>
<td>SOFA</td>
<td>EAST</td>
<td>842.00</td>
<td>988.00</td>
</tr>
<tr>
<td>7</td>
<td>SOFA</td>
<td>EAST</td>
<td>812.00</td>
<td>717.00</td>
</tr>
<tr>
<td>8</td>
<td>SOFA</td>
<td>EAST</td>
<td>811.00</td>
<td>504.00</td>
</tr>
</tbody>
</table>
Output 14.18  Meadow Style

![Bar chart showing sales for different furniture types in the East and West regions.](image1)

![Scatter plot showing the relationship between height and weight with 95% confidence limits and prediction limits.](image2)

<table>
<thead>
<tr>
<th>Obs</th>
<th>PRODUCT</th>
<th>REGION</th>
<th>ACTUAL</th>
<th>PREDICT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>SOFA</td>
<td>EAST</td>
<td>$925.00</td>
<td>$590.00</td>
</tr>
<tr>
<td>2</td>
<td>SOFA</td>
<td>EAST</td>
<td>$900.00</td>
<td>$207.00</td>
</tr>
<tr>
<td>3</td>
<td>SOFA</td>
<td>EAST</td>
<td>$988.00</td>
<td>$846.00</td>
</tr>
<tr>
<td>4</td>
<td>SOFA</td>
<td>EAST</td>
<td>$542.00</td>
<td>$2533.00</td>
</tr>
<tr>
<td>5</td>
<td>SOFA</td>
<td>EAST</td>
<td>$908.00</td>
<td>$846.00</td>
</tr>
<tr>
<td>6</td>
<td>SOFA</td>
<td>EAST</td>
<td>$946.00</td>
<td>$496.00</td>
</tr>
<tr>
<td>7</td>
<td>SOFA</td>
<td>EAST</td>
<td>$812.00</td>
<td>$717.00</td>
</tr>
<tr>
<td>8</td>
<td>SOFA</td>
<td>EAST</td>
<td>$114.00</td>
<td>$584.00</td>
</tr>
</tbody>
</table>
Output 14.19  Minimal Style

![Bar charts and scatter plot with data table]

**Table 1:**

<table>
<thead>
<tr>
<th>Obs</th>
<th>PRODUCT</th>
<th>REGION</th>
<th>ACTUAL</th>
<th>PREDICT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>SOFA</td>
<td>EAST</td>
<td>$925.00</td>
<td>$850.00</td>
</tr>
<tr>
<td>2</td>
<td>SOFA</td>
<td>EAST</td>
<td>$999.00</td>
<td>$937.00</td>
</tr>
<tr>
<td>3</td>
<td>SOFA</td>
<td>EAST</td>
<td>$608.00</td>
<td>$846.00</td>
</tr>
<tr>
<td>4</td>
<td>SOFA</td>
<td>EAST</td>
<td>$642.00</td>
<td>$533.00</td>
</tr>
<tr>
<td>5</td>
<td>SOFA</td>
<td>EAST</td>
<td>$656.00</td>
<td>$646.00</td>
</tr>
<tr>
<td>6</td>
<td>SOFA</td>
<td>EAST</td>
<td>$948.00</td>
<td>$486.00</td>
</tr>
<tr>
<td>7</td>
<td>SOFA</td>
<td>EAST</td>
<td>$612.00</td>
<td>$717.00</td>
</tr>
<tr>
<td>8</td>
<td>SOFA</td>
<td>EAST</td>
<td>$114.00</td>
<td>$564.00</td>
</tr>
</tbody>
</table>
Output 14.20  Netdraw and NoFontDefault Styles

![Graph showing predicted and actual sales for different regions.](image)

![Graph showing regression analysis between weight and height.](image)

<table>
<thead>
<tr>
<th>Obs</th>
<th>PRODUCT</th>
<th>REGION</th>
<th>ACTUAL</th>
<th>PREDICT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>SOFA</td>
<td>EAST</td>
<td>$925.00</td>
<td>$850.00</td>
</tr>
<tr>
<td>2</td>
<td>SOFA</td>
<td>EAST</td>
<td>$999.00</td>
<td>$297.00</td>
</tr>
<tr>
<td>3</td>
<td>SOFA</td>
<td>EAST</td>
<td>$608.00</td>
<td>$846.00</td>
</tr>
<tr>
<td>4</td>
<td>SOFA</td>
<td>EAST</td>
<td>$642.00</td>
<td>$533.00</td>
</tr>
<tr>
<td>5</td>
<td>SOFA</td>
<td>EAST</td>
<td>$656.00</td>
<td>$846.00</td>
</tr>
<tr>
<td>6</td>
<td>SOFA</td>
<td>EAST</td>
<td>$948.00</td>
<td>$466.00</td>
</tr>
<tr>
<td>7</td>
<td>SOFA</td>
<td>EAST</td>
<td>$612.00</td>
<td>$717.00</td>
</tr>
<tr>
<td>8</td>
<td>SOFA</td>
<td>EAST</td>
<td>$444.22</td>
<td>$564.22</td>
</tr>
</tbody>
</table>
Output 14.21  Normal Style

<table>
<thead>
<tr>
<th>Region</th>
<th>BED</th>
<th>CHAIR</th>
<th>DESK</th>
</tr>
</thead>
<tbody>
<tr>
<td>EAST</td>
<td>$\downarrow$</td>
<td>$\downarrow$</td>
<td>$\downarrow$</td>
</tr>
<tr>
<td>WEST</td>
<td>$\downarrow$</td>
<td>$\downarrow$</td>
<td>$\downarrow$</td>
</tr>
</tbody>
</table>

Sales

- $0
- $5,000
- $10,000
- $15,000
- $20,000

- $0
- $5,000
- $10,000
- $15,000
- $20,000

- $0
- $5,000
- $10,000
- $15,000
- $20,000

- $0
- $5,000
- $10,000
- $15,000
- $20,000

<table>
<thead>
<tr>
<th>Region</th>
<th>SOFA</th>
<th>TABLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>EAST</td>
<td>$\downarrow$</td>
<td>$\downarrow$</td>
</tr>
<tr>
<td>WEST</td>
<td>$\downarrow$</td>
<td>$\downarrow$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Obs</th>
<th>PRODUCT</th>
<th>REGION</th>
<th>ACTUAL</th>
<th>PREDICT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>SOFA</td>
<td>EAST</td>
<td>$206.00</td>
<td>$250.00</td>
</tr>
<tr>
<td>2</td>
<td>SOFA</td>
<td>EAST</td>
<td>$206.00</td>
<td>$250.00</td>
</tr>
<tr>
<td>3</td>
<td>SOFA</td>
<td>EAST</td>
<td>$206.00</td>
<td>$250.00</td>
</tr>
<tr>
<td>4</td>
<td>SOFA</td>
<td>EAST</td>
<td>$206.00</td>
<td>$250.00</td>
</tr>
<tr>
<td>5</td>
<td>SOFA</td>
<td>EAST</td>
<td>$206.00</td>
<td>$250.00</td>
</tr>
<tr>
<td>6</td>
<td>SOFA</td>
<td>EAST</td>
<td>$206.00</td>
<td>$250.00</td>
</tr>
<tr>
<td>7</td>
<td>SOFA</td>
<td>EAST</td>
<td>$206.00</td>
<td>$250.00</td>
</tr>
<tr>
<td>8</td>
<td>SOFA</td>
<td>EAST</td>
<td>$206.00</td>
<td>$250.00</td>
</tr>
</tbody>
</table>
Output 14.22  Ocean Style

[Graphs and tables related to sales and weight predictions]
Output 14.23  Plateau Style

| ODS Styles Gallery | 297 |

### Chart 1: Sales by Region and Product

<table>
<thead>
<tr>
<th>Region</th>
<th>Bed</th>
<th>Chair</th>
<th>Desk</th>
</tr>
</thead>
<tbody>
<tr>
<td>East</td>
<td>$20,000</td>
<td>$15,000</td>
<td>$10,000</td>
</tr>
<tr>
<td>West</td>
<td>$10,000</td>
<td>$15,000</td>
<td>$20,000</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Product</th>
<th>Sofa</th>
<th>Table</th>
</tr>
</thead>
<tbody>
<tr>
<td>East</td>
<td>$10,000</td>
<td>$15,000</td>
</tr>
<tr>
<td>West</td>
<td>$15,000</td>
<td>$20,000</td>
</tr>
</tbody>
</table>

### Chart 2: Weight vs. Height with Regression Analysis

- 95% Confidence Limits
- 95% Prediction Limits
- Regression Line

<table>
<thead>
<tr>
<th>Obs</th>
<th>Product</th>
<th>Region</th>
<th>Actual</th>
<th>Predict</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Sofa</td>
<td>East</td>
<td>$920.00</td>
<td>$800.00</td>
</tr>
<tr>
<td>2</td>
<td>Sofa</td>
<td>East</td>
<td>$666.00</td>
<td>$297.00</td>
</tr>
<tr>
<td>3</td>
<td>Sofa</td>
<td>East</td>
<td>$600.00</td>
<td>$948.00</td>
</tr>
<tr>
<td>4</td>
<td>Sofa</td>
<td>East</td>
<td>$542.00</td>
<td>$533.00</td>
</tr>
<tr>
<td>5</td>
<td>Sofa</td>
<td>East</td>
<td>$566.00</td>
<td>$546.00</td>
</tr>
<tr>
<td>6</td>
<td>Sofa</td>
<td>East</td>
<td>$448.00</td>
<td>$428.00</td>
</tr>
<tr>
<td>7</td>
<td>Sofa</td>
<td>East</td>
<td>$512.00</td>
<td>$717.00</td>
</tr>
<tr>
<td>8</td>
<td>Sofa</td>
<td>East</td>
<td>$114.00</td>
<td>$504.00</td>
</tr>
</tbody>
</table>
Output 14.24  Raven Style

![Graph showing sales comparison between regions for different products]

- **BED**
  - East: $18,500
  - West: $19,000

- **CHAIR**
  - East: $20,000
  - West: $21,500

- **DESK**
  - East: $17,000
  - West: $18,500

- **SOFA**
  - East: $22,000
  - West: $23,000

- **TABLE**
  - East: $15,000
  - West: $16,500

---

![Graph showing weight vs. height with regression line and confidence limits]

- **Confidence Limits**
- **Prediction Limits**
- **Regression Line**

---

<table>
<thead>
<tr>
<th>Obs</th>
<th>PRODUCT</th>
<th>REGION</th>
<th>ACTUAL</th>
<th>PREDICT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>SOFA</td>
<td>EAST</td>
<td>$625.00</td>
<td>$550.00</td>
</tr>
<tr>
<td>2</td>
<td>SOFA</td>
<td>EAST</td>
<td>$600.00</td>
<td>$537.00</td>
</tr>
<tr>
<td>3</td>
<td>SOFA</td>
<td>EAST</td>
<td>$650.00</td>
<td>$546.00</td>
</tr>
<tr>
<td>4</td>
<td>SOFA</td>
<td>EAST</td>
<td>$640.00</td>
<td>$533.00</td>
</tr>
<tr>
<td>5</td>
<td>SOFA</td>
<td>EAST</td>
<td>$605.00</td>
<td>$540.00</td>
</tr>
<tr>
<td>6</td>
<td>SOFA</td>
<td>EAST</td>
<td>$640.00</td>
<td>$490.00</td>
</tr>
<tr>
<td>7</td>
<td>SOFA</td>
<td>EAST</td>
<td>$612.00</td>
<td>$717.00</td>
</tr>
<tr>
<td>8</td>
<td>SOFA</td>
<td>EAST</td>
<td>$114.00</td>
<td>$594.00</td>
</tr>
</tbody>
</table>
Output 14.25  SasWeb Style

---

**Sales by Region**

- **Region:** EAST, WEST
- **Products:** BED, CHAIR, DESK, SOFA, TABLE
- **Sales Range:** $0 to $20,000

---

**Weight vs. Height**

- **Graph** shows a scatter plot with lines indicating 95% confidence and prediction limits.
- **Regression line** is also visible.

---

**Obs | PRODUCT | REGION | ACTUAL | PREDICT**
--- | --- | --- | --- | ---
1 | SOFA | EAST | $625.00 | $650.00
2 | SOFA | EAST | $600.00 | $597.00
3 | SOFA | EAST | $609.00 | $646.00
4 | SOFA | EAST | $542.00 | $533.00
5 | SOFA | EAST | $688.00 | $846.00
6 | SOFA | EAST | $848.00 | $490.00
7 | SOFA | EAST | $512.00 | $717.00
8 | SOFA | EAST | $314.00 | $504.00
Output 14.26  Seaside Style
Output 14.27  StatDoc Style

- Bar chart showing sales by region for different products (Bed, Chair, Desk, Sofa, Table) in two regions (East, West).
- The chart includes predicted sales and actual sales.

- Scatter plot showing the relationship between height and weight, with 95% confidence limits, 95% prediction limits, and regression line.

- Table showingObs, PRODUCT, REGION, ACTUAL, PREDICT:

<table>
<thead>
<tr>
<th>Obs</th>
<th>PRODUCT</th>
<th>REGION</th>
<th>ACTUAL</th>
<th>PREDICT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>SOFA</td>
<td>EAST</td>
<td>$620.00</td>
<td>$561.00</td>
</tr>
<tr>
<td>2</td>
<td>SOFA</td>
<td>EAST</td>
<td>$600.00</td>
<td>$257.00</td>
</tr>
<tr>
<td>3</td>
<td>SOFA</td>
<td>EAST</td>
<td>$603.00</td>
<td>$846.00</td>
</tr>
<tr>
<td>4</td>
<td>SOFA</td>
<td>EAST</td>
<td>$642.00</td>
<td>$632.00</td>
</tr>
<tr>
<td>5</td>
<td>SOFA</td>
<td>EAST</td>
<td>$300.00</td>
<td>$640.00</td>
</tr>
<tr>
<td>6</td>
<td>SOFA</td>
<td>EAST</td>
<td>$482.00</td>
<td>$486.00</td>
</tr>
<tr>
<td>7</td>
<td>SOFA</td>
<td>EAST</td>
<td>$812.00</td>
<td>$717.00</td>
</tr>
<tr>
<td>8</td>
<td>SOFA</td>
<td>EAST</td>
<td>$114.00</td>
<td>$564.00</td>
</tr>
</tbody>
</table>
Output 14.28 Statistical Style

**Table:**

<table>
<thead>
<tr>
<th>Obs</th>
<th>PRODUCT</th>
<th>REGION</th>
<th>ACTUAL</th>
<th>PREDICT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>SOFA</td>
<td>EAST</td>
<td>$925.00</td>
<td>$850.00</td>
</tr>
<tr>
<td>2</td>
<td>SOFA</td>
<td>EAST</td>
<td>$999.00</td>
<td>$287.00</td>
</tr>
<tr>
<td>3</td>
<td>SOFA</td>
<td>EAST</td>
<td>$606.00</td>
<td>$646.00</td>
</tr>
<tr>
<td>4</td>
<td>SOFA</td>
<td>EAST</td>
<td>$642.00</td>
<td>$533.00</td>
</tr>
<tr>
<td>5</td>
<td>SOFA</td>
<td>EAST</td>
<td>$550.00</td>
<td>$646.00</td>
</tr>
<tr>
<td>6</td>
<td>SOFA</td>
<td>EAST</td>
<td>$948.00</td>
<td>$466.00</td>
</tr>
<tr>
<td>7</td>
<td>SOFA</td>
<td>EAST</td>
<td>$869.00</td>
<td>$747.00</td>
</tr>
</tbody>
</table>
### Printer Styles

**Output 14.29  FancyPrinter Style**

<table>
<thead>
<tr>
<th>Obs</th>
<th>Make</th>
<th>Model</th>
<th>Type</th>
<th>Origin</th>
<th>Invoice</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Acura</td>
<td>MDX</td>
<td>SUV</td>
<td>Asia</td>
<td>$33,337</td>
</tr>
<tr>
<td>2</td>
<td>Acura</td>
<td>RSX Type S 2dr</td>
<td>Sedan</td>
<td>Asia</td>
<td>$21,761</td>
</tr>
<tr>
<td>3</td>
<td>Acura</td>
<td>TSX 4dr</td>
<td>Sedan</td>
<td>Asia</td>
<td>$24,647</td>
</tr>
<tr>
<td>4</td>
<td>Acura</td>
<td>TL 4dr</td>
<td>Sedan</td>
<td>Asia</td>
<td>$30,299</td>
</tr>
<tr>
<td>5</td>
<td>Acura</td>
<td>3.5 RL 4dr</td>
<td>Sedan</td>
<td>Asia</td>
<td>$39,014</td>
</tr>
<tr>
<td>6</td>
<td>Acura</td>
<td>3.5 RL w/Navigation 4dr</td>
<td>Sedan</td>
<td>Asia</td>
<td>$41,100</td>
</tr>
<tr>
<td>7</td>
<td>Acura</td>
<td>NSX coupe 2dr manual S</td>
<td>Sports</td>
<td>Asia</td>
<td>$79,978</td>
</tr>
<tr>
<td>8</td>
<td>Audi</td>
<td>A4 1.8T 4dr</td>
<td>Sedan</td>
<td>Europe</td>
<td>$23,508</td>
</tr>
<tr>
<td>9</td>
<td>Audi</td>
<td>A4 1.8T convertible 2dr</td>
<td>Sedan</td>
<td>Europe</td>
<td>$32,506</td>
</tr>
<tr>
<td>10</td>
<td>Audi</td>
<td>A4 3.0 4dr</td>
<td>Sedan</td>
<td>Europe</td>
<td>$28,846</td>
</tr>
<tr>
<td>11</td>
<td>Audi</td>
<td>A4 3.0 Quattro 4dr manual</td>
<td>Sedan</td>
<td>Europe</td>
<td>$30,366</td>
</tr>
<tr>
<td>12</td>
<td>Audi</td>
<td>A4 3.0 Quattro 4dr auto</td>
<td>Sedan</td>
<td>Europe</td>
<td>$31,388</td>
</tr>
<tr>
<td>13</td>
<td>Audi</td>
<td>A6 3.0 4dr</td>
<td>Sedan</td>
<td>Europe</td>
<td>$33,129</td>
</tr>
<tr>
<td>14</td>
<td>Audi</td>
<td>A6 3.0 Quattro 4dr</td>
<td>Sedan</td>
<td>Europe</td>
<td>$35,992</td>
</tr>
<tr>
<td>15</td>
<td>Audi</td>
<td>A4 3.0 convertible 2dr</td>
<td>Sedan</td>
<td>Europe</td>
<td>$38,325</td>
</tr>
</tbody>
</table>

**Output 14.30  FestivalPrinter Style**

<table>
<thead>
<tr>
<th>Obs</th>
<th>Make</th>
<th>Model</th>
<th>Type</th>
<th>Origin</th>
<th>Invoice</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Acura</td>
<td>MDX</td>
<td>SUV</td>
<td>Asia</td>
<td>$33,337</td>
</tr>
<tr>
<td>2</td>
<td>Acura</td>
<td>RSX Type S 2dr</td>
<td>Sedan</td>
<td>Asia</td>
<td>$21,761</td>
</tr>
<tr>
<td>3</td>
<td>Acura</td>
<td>TSX 4dr</td>
<td>Sedan</td>
<td>Asia</td>
<td>$24,647</td>
</tr>
<tr>
<td>4</td>
<td>Acura</td>
<td>TL 4dr</td>
<td>Sedan</td>
<td>Asia</td>
<td>$30,299</td>
</tr>
<tr>
<td>5</td>
<td>Acura</td>
<td>3.5 RL 4dr</td>
<td>Sedan</td>
<td>Asia</td>
<td>$39,014</td>
</tr>
<tr>
<td>6</td>
<td>Acura</td>
<td>3.5 RL w/Navigation 4dr</td>
<td>Sedan</td>
<td>Asia</td>
<td>$41,100</td>
</tr>
<tr>
<td>7</td>
<td>Acura</td>
<td>NSX coupe 2dr manual S</td>
<td>Sports</td>
<td>Asia</td>
<td>$79,978</td>
</tr>
<tr>
<td>8</td>
<td>Audi</td>
<td>A4 1.8T 4dr</td>
<td>Sedan</td>
<td>Europe</td>
<td>$23,508</td>
</tr>
<tr>
<td>9</td>
<td>Audi</td>
<td>A4 1.8T convertible 2dr</td>
<td>Sedan</td>
<td>Europe</td>
<td>$32,506</td>
</tr>
<tr>
<td>10</td>
<td>Audi</td>
<td>A4 3.0 4dr</td>
<td>Sedan</td>
<td>Europe</td>
<td>$28,846</td>
</tr>
<tr>
<td>11</td>
<td>Audi</td>
<td>A4 3.0 Quattro 4dr manual</td>
<td>Sedan</td>
<td>Europe</td>
<td>$30,366</td>
</tr>
<tr>
<td>12</td>
<td>Audi</td>
<td>A4 3.0 Quattro 4dr auto</td>
<td>Sedan</td>
<td>Europe</td>
<td>$31,388</td>
</tr>
<tr>
<td>13</td>
<td>Audi</td>
<td>A6 3.0 4dr</td>
<td>Sedan</td>
<td>Europe</td>
<td>$33,129</td>
</tr>
<tr>
<td>14</td>
<td>Audi</td>
<td>A6 3.0 Quattro 4dr</td>
<td>Sedan</td>
<td>Europe</td>
<td>$35,992</td>
</tr>
<tr>
<td>15</td>
<td>Audi</td>
<td>A4 3.0 convertible 2dr</td>
<td>Sedan</td>
<td>Europe</td>
<td>$38,325</td>
</tr>
</tbody>
</table>
### Output 14.31 GrayscalePrinter Style

<table>
<thead>
<tr>
<th>Obs</th>
<th>Make</th>
<th>Model</th>
<th>Type</th>
<th>Origin</th>
<th>Invoice</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Acura</td>
<td>MDX</td>
<td>SUV</td>
<td>Asia</td>
<td>$33,337</td>
</tr>
<tr>
<td>2</td>
<td>Acura</td>
<td>RSX Type S 2dr</td>
<td>Sedan</td>
<td>Asia</td>
<td>$21,761</td>
</tr>
<tr>
<td>3</td>
<td>Acura</td>
<td>TSX 4dr</td>
<td>Sedan</td>
<td>Asia</td>
<td>$24,647</td>
</tr>
<tr>
<td>4</td>
<td>Acura</td>
<td>TL 4dr</td>
<td>Sedan</td>
<td>Asia</td>
<td>$30,299</td>
</tr>
<tr>
<td>5</td>
<td>Acura</td>
<td>3.5 RL 4dr</td>
<td>Sedan</td>
<td>Asia</td>
<td>$39,014</td>
</tr>
<tr>
<td>6</td>
<td>Acura</td>
<td>3.5 RL w/Navigation 4dr</td>
<td>Sedan</td>
<td>Asia</td>
<td>$41,100</td>
</tr>
<tr>
<td>7</td>
<td>Acura</td>
<td>NSX coupe 2dr manual S</td>
<td>Sports</td>
<td>Asia</td>
<td>$79,978</td>
</tr>
<tr>
<td>8</td>
<td>Audi</td>
<td>A4 1.8T 4dr</td>
<td>Sedan</td>
<td>Europe</td>
<td>$23,508</td>
</tr>
<tr>
<td>9</td>
<td>Audi</td>
<td>A4 1.8T convertible 2dr</td>
<td>Sedan</td>
<td>Europe</td>
<td>$32,506</td>
</tr>
<tr>
<td>10</td>
<td>Audi</td>
<td>A4 3.0 4dr</td>
<td>Sedan</td>
<td>Europe</td>
<td>$28,846</td>
</tr>
<tr>
<td>11</td>
<td>Audi</td>
<td>A4 3.0 Quattro 4dr manual</td>
<td>Sedan</td>
<td>Europe</td>
<td>$30,306</td>
</tr>
<tr>
<td>12</td>
<td>Audi</td>
<td>A4 3.0 Quattro 4dr auto</td>
<td>Sedan</td>
<td>Europe</td>
<td>$31,388</td>
</tr>
<tr>
<td>13</td>
<td>Audi</td>
<td>A6 3.0 4dr</td>
<td>Sedan</td>
<td>Europe</td>
<td>$33,129</td>
</tr>
<tr>
<td>14</td>
<td>Audi</td>
<td>A6 3.0 Quattro 4dr</td>
<td>Sedan</td>
<td>Europe</td>
<td>$35,992</td>
</tr>
<tr>
<td>15</td>
<td>Audi</td>
<td>A4 3.0 convertible 2dr</td>
<td>Sedan</td>
<td>Europe</td>
<td>$38,325</td>
</tr>
</tbody>
</table>

### Output 14.32 MeadowPrinter Style

<table>
<thead>
<tr>
<th>Obs</th>
<th>Make</th>
<th>Model</th>
<th>Type</th>
<th>Origin</th>
<th>Invoice</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Acura</td>
<td>MDX</td>
<td>SUV</td>
<td>Asia</td>
<td>$33,337</td>
</tr>
<tr>
<td>2</td>
<td>Acura</td>
<td>RSX Type S 2dr</td>
<td>Sedan</td>
<td>Asia</td>
<td>$21,761</td>
</tr>
<tr>
<td>3</td>
<td>Acura</td>
<td>TSX 4dr</td>
<td>Sedan</td>
<td>Asia</td>
<td>$24,647</td>
</tr>
<tr>
<td>4</td>
<td>Acura</td>
<td>TL 4dr</td>
<td>Sedan</td>
<td>Asia</td>
<td>$30,299</td>
</tr>
<tr>
<td>5</td>
<td>Acura</td>
<td>3.5 RL 4dr</td>
<td>Sedan</td>
<td>Asia</td>
<td>$39,014</td>
</tr>
<tr>
<td>6</td>
<td>Acura</td>
<td>3.5 RL w/Navigation 4dr</td>
<td>Sedan</td>
<td>Asia</td>
<td>$41,100</td>
</tr>
<tr>
<td>7</td>
<td>Acura</td>
<td>NSX coupe 2dr manual S</td>
<td>Sports</td>
<td>Asia</td>
<td>$79,978</td>
</tr>
<tr>
<td>8</td>
<td>Audi</td>
<td>A4 1.8T 4dr</td>
<td>Sedan</td>
<td>Europe</td>
<td>$23,508</td>
</tr>
<tr>
<td>9</td>
<td>Audi</td>
<td>A4 1.8T convertible 2dr</td>
<td>Sedan</td>
<td>Europe</td>
<td>$32,506</td>
</tr>
<tr>
<td>10</td>
<td>Audi</td>
<td>A4 3.0 4dr</td>
<td>Sedan</td>
<td>Europe</td>
<td>$28,846</td>
</tr>
<tr>
<td>11</td>
<td>Audi</td>
<td>A4 3.0 Quattro 4dr manual</td>
<td>Sedan</td>
<td>Europe</td>
<td>$30,306</td>
</tr>
<tr>
<td>12</td>
<td>Audi</td>
<td>A4 3.0 Quattro 4dr auto</td>
<td>Sedan</td>
<td>Europe</td>
<td>$31,388</td>
</tr>
<tr>
<td>13</td>
<td>Audi</td>
<td>A6 3.0 4dr</td>
<td>Sedan</td>
<td>Europe</td>
<td>$33,129</td>
</tr>
<tr>
<td>14</td>
<td>Audi</td>
<td>A6 3.0 Quattro 4dr</td>
<td>Sedan</td>
<td>Europe</td>
<td>$35,992</td>
</tr>
<tr>
<td>15</td>
<td>Audi</td>
<td>A4 3.0 convertible 2dr</td>
<td>Sedan</td>
<td>Europe</td>
<td>$38,325</td>
</tr>
</tbody>
</table>
### Output 14.33  MonoChromePrinter Style

<table>
<thead>
<tr>
<th>Obs</th>
<th>Make</th>
<th>Model</th>
<th>Type</th>
<th>Origin</th>
<th>Invoice</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Acura</td>
<td>MDX</td>
<td>SUV</td>
<td>Asia</td>
<td>$33,337</td>
</tr>
<tr>
<td>2</td>
<td>Acura</td>
<td>RSX Type S 2dr</td>
<td>Sedan</td>
<td>Asia</td>
<td>$21,761</td>
</tr>
<tr>
<td>3</td>
<td>Acura</td>
<td>TSX 4dr</td>
<td>Sedan</td>
<td>Asia</td>
<td>$24,647</td>
</tr>
<tr>
<td>4</td>
<td>Acura</td>
<td>TL 4dr</td>
<td>Sedan</td>
<td>Asia</td>
<td>$30,299</td>
</tr>
<tr>
<td>5</td>
<td>Acura</td>
<td>3.5 RL 4dr</td>
<td>Sedan</td>
<td>Asia</td>
<td>$39,014</td>
</tr>
<tr>
<td>6</td>
<td>Acura</td>
<td>3.5 RL w/Navigation 4dr</td>
<td>Sedan</td>
<td>Asia</td>
<td>$41,100</td>
</tr>
<tr>
<td>7</td>
<td>Acura</td>
<td>NSX coupe 2dr manual S</td>
<td>Sports</td>
<td>Asia</td>
<td>$79,978</td>
</tr>
<tr>
<td>8</td>
<td>Audi</td>
<td>A4 1.8T 4dr</td>
<td>Sedan</td>
<td>Europe</td>
<td>$23,508</td>
</tr>
<tr>
<td>9</td>
<td>Audi</td>
<td>A4 1.8T convertible 2dr</td>
<td>Sedan</td>
<td>Europe</td>
<td>$22,506</td>
</tr>
<tr>
<td>10</td>
<td>Audi</td>
<td>A4 3.0 4dr</td>
<td>Sedan</td>
<td>Europe</td>
<td>$38,846</td>
</tr>
<tr>
<td>11</td>
<td>Audi</td>
<td>A4 3.0 Quattro 4dr manual</td>
<td>Sedan</td>
<td>Europe</td>
<td>$30,366</td>
</tr>
<tr>
<td>12</td>
<td>Audi</td>
<td>A4 3.0 Quattro 4dr auto</td>
<td>Sedan</td>
<td>Europe</td>
<td>$31,388</td>
</tr>
<tr>
<td>13</td>
<td>Audi</td>
<td>A6 3.0 4dr</td>
<td>Sedan</td>
<td>Europe</td>
<td>$33,129</td>
</tr>
<tr>
<td>14</td>
<td>Audi</td>
<td>A6 3.0 Quattro 4dr</td>
<td>Sedan</td>
<td>Europe</td>
<td>$35,992</td>
</tr>
<tr>
<td>15</td>
<td>Audi</td>
<td>A6 3.0 convertible 2dr</td>
<td>Sedan</td>
<td>Europe</td>
<td>$38,325</td>
</tr>
</tbody>
</table>

### Output 14.34  Monospace Style

<table>
<thead>
<tr>
<th>Obs</th>
<th>Make</th>
<th>Model</th>
<th>Type</th>
<th>Origin</th>
<th>Invoice</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Acura</td>
<td>MDX</td>
<td>SUV</td>
<td>Asia</td>
<td>$33,337</td>
</tr>
<tr>
<td>2</td>
<td>Acura</td>
<td>RSX Type S 2dr</td>
<td>Sedan</td>
<td>Asia</td>
<td>$21,761</td>
</tr>
<tr>
<td>3</td>
<td>Acura</td>
<td>TSX 4dr</td>
<td>Sedan</td>
<td>Asia</td>
<td>$24,647</td>
</tr>
<tr>
<td>4</td>
<td>Acura</td>
<td>TL 4dr</td>
<td>Sedan</td>
<td>Asia</td>
<td>$30,299</td>
</tr>
<tr>
<td>5</td>
<td>Acura</td>
<td>3.5 RL 4dr</td>
<td>Sedan</td>
<td>Asia</td>
<td>$39,014</td>
</tr>
<tr>
<td>6</td>
<td>Acura</td>
<td>3.5 RL w/Navigation 4dr</td>
<td>Sedan</td>
<td>Asia</td>
<td>$41,100</td>
</tr>
<tr>
<td>7</td>
<td>Acura</td>
<td>NSX coupe 2dr manual S</td>
<td>Sports</td>
<td>Asia</td>
<td>$79,978</td>
</tr>
<tr>
<td>8</td>
<td>Audi</td>
<td>A4 1.8T 4dr</td>
<td>Sedan</td>
<td>Europe</td>
<td>$23,508</td>
</tr>
<tr>
<td>9</td>
<td>Audi</td>
<td>A4 1.8T convertible 2dr</td>
<td>Sedan</td>
<td>Europe</td>
<td>$22,506</td>
</tr>
<tr>
<td>10</td>
<td>Audi</td>
<td>A4 3.0 4dr</td>
<td>Sedan</td>
<td>Europe</td>
<td>$38,846</td>
</tr>
<tr>
<td>11</td>
<td>Audi</td>
<td>A4 3.0 Quattro 4dr manual</td>
<td>Sedan</td>
<td>Europe</td>
<td>$30,366</td>
</tr>
<tr>
<td>12</td>
<td>Audi</td>
<td>A4 3.0 Quattro 4dr auto</td>
<td>Sedan</td>
<td>Europe</td>
<td>$31,388</td>
</tr>
<tr>
<td>13</td>
<td>Audi</td>
<td>A6 3.0 4dr</td>
<td>Sedan</td>
<td>Europe</td>
<td>$33,129</td>
</tr>
<tr>
<td>14</td>
<td>Audi</td>
<td>A6 3.0 Quattro 4dr</td>
<td>Sedan</td>
<td>Europe</td>
<td>$35,992</td>
</tr>
<tr>
<td>15</td>
<td>Audi</td>
<td>A6 3.0 convertible 2dr</td>
<td>Sedan</td>
<td>Europe</td>
<td>$38,325</td>
</tr>
</tbody>
</table>
Pearl is the default style for PRINTER output.

Output 14.35  NormalPrinter Style

Output 14.36  Pearl Style
### Output 14.37  Sapphire Style

<table>
<thead>
<tr>
<th>Obs</th>
<th>Make</th>
<th>Model</th>
<th>Type</th>
<th>Origin</th>
<th>Invoice</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Acura</td>
<td>MDX</td>
<td>SUV</td>
<td>Asia</td>
<td>$33,337</td>
</tr>
<tr>
<td>2</td>
<td>Acura</td>
<td>RSX Type S 2dr</td>
<td>Sedan</td>
<td>Asia</td>
<td>$21,761</td>
</tr>
<tr>
<td>3</td>
<td>Acura</td>
<td>TSX 4dr</td>
<td>Sedan</td>
<td>Asia</td>
<td>$24,647</td>
</tr>
<tr>
<td>4</td>
<td>Acura</td>
<td>TL 4dr</td>
<td>Sedan</td>
<td>Asia</td>
<td>$30,299</td>
</tr>
<tr>
<td>5</td>
<td>Acura</td>
<td>3.5 RL 4dr</td>
<td>Sedan</td>
<td>Asia</td>
<td>$39,014</td>
</tr>
<tr>
<td>6</td>
<td>Acura</td>
<td>3.5 RL w/Navigation 4dr</td>
<td>Sedan</td>
<td>Asia</td>
<td>$41,100</td>
</tr>
<tr>
<td>7</td>
<td>Acura</td>
<td>NSX coupe 2dr manual S</td>
<td>Sports</td>
<td>Asia</td>
<td>$79,978</td>
</tr>
<tr>
<td>8</td>
<td>Audi</td>
<td>A4 1.8T 4dr</td>
<td>Sedan</td>
<td>Europe</td>
<td>$23,508</td>
</tr>
<tr>
<td>9</td>
<td>Audi</td>
<td>A4 1.8T convertible 2dr</td>
<td>Sedan</td>
<td>Europe</td>
<td>$32,506</td>
</tr>
<tr>
<td>10</td>
<td>Audi</td>
<td>A4 3.0 4dr</td>
<td>Sedan</td>
<td>Europe</td>
<td>$28,846</td>
</tr>
<tr>
<td>11</td>
<td>Audi</td>
<td>A4 3.0 Quattro 4dr manual</td>
<td>Sedan</td>
<td>Europe</td>
<td>$30,366</td>
</tr>
<tr>
<td>12</td>
<td>Audi</td>
<td>A4 3.0 Quattro 4dr auto</td>
<td>Sedan</td>
<td>Europe</td>
<td>$31,388</td>
</tr>
<tr>
<td>13</td>
<td>Audi</td>
<td>A6 3.0 4dr</td>
<td>Sedan</td>
<td>Europe</td>
<td>$33,129</td>
</tr>
<tr>
<td>14</td>
<td>Audi</td>
<td>A6 3.0 Quattro 4dr</td>
<td>Sedan</td>
<td>Europe</td>
<td>$35,992</td>
</tr>
<tr>
<td>15</td>
<td>Audi</td>
<td>A4 3.0 convertible 2dr</td>
<td>Sedan</td>
<td>Europe</td>
<td>$38,325</td>
</tr>
</tbody>
</table>

### Output 14.38  SasDocPrinter Style
Output 14.39  **Seaside Printer Style**

<table>
<thead>
<tr>
<th>Obs</th>
<th>Make</th>
<th>Model</th>
<th>Type</th>
<th>Origin</th>
<th>Invoice</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Acura</td>
<td>MDX</td>
<td>SUV</td>
<td>Asia</td>
<td>$33,337</td>
</tr>
<tr>
<td>2</td>
<td>Acura</td>
<td>RXS Type S 2dr</td>
<td>Sedan</td>
<td>Asia</td>
<td>$21,761</td>
</tr>
<tr>
<td>3</td>
<td>Acura</td>
<td>TSX 4dr</td>
<td>Sedan</td>
<td>Asia</td>
<td>$24,647</td>
</tr>
<tr>
<td>4</td>
<td>Acura</td>
<td>TL 4dr</td>
<td>Sedan</td>
<td>Asia</td>
<td>$30,299</td>
</tr>
<tr>
<td>5</td>
<td>Acura</td>
<td>3.5 RL 4dr</td>
<td>Sedan</td>
<td>Asia</td>
<td>$36,014</td>
</tr>
<tr>
<td>6</td>
<td>Acura</td>
<td>3.5 RL w/Nav 4dr</td>
<td>Sedan</td>
<td>Asia</td>
<td>$41,100</td>
</tr>
<tr>
<td>7</td>
<td>Acura</td>
<td>NSX Coupe 2dr manual S</td>
<td>Sports</td>
<td>Asia</td>
<td>$79,978</td>
</tr>
<tr>
<td>8</td>
<td>Audi</td>
<td>A4 1.8T 4dr</td>
<td>Sedan</td>
<td>Europe</td>
<td>$23,528</td>
</tr>
<tr>
<td>9</td>
<td>Audi</td>
<td>A4 1.8T convertible 2dr</td>
<td>Sedan</td>
<td>Europe</td>
<td>$32,506</td>
</tr>
<tr>
<td>10</td>
<td>Audi</td>
<td>A4 3.0 4dr</td>
<td>Sedan</td>
<td>Europe</td>
<td>$29,846</td>
</tr>
<tr>
<td>11</td>
<td>Audi</td>
<td>A4 3.0 Quattro 4dr manual</td>
<td>Sedan</td>
<td>Europe</td>
<td>$30,366</td>
</tr>
<tr>
<td>12</td>
<td>Audi</td>
<td>A4 3.0 Quattro 4dr auto</td>
<td>Sedan</td>
<td>Europe</td>
<td>$31,388</td>
</tr>
<tr>
<td>13</td>
<td>Audi</td>
<td>A6 3.0 4dr</td>
<td>Sedan</td>
<td>Europe</td>
<td>$33,129</td>
</tr>
</tbody>
</table>

**Styles for the ODS Destination for PowerPoint**

PowerPointLight is the default style for output created by the ODS destination for PowerPoint.

Output 14.40  **PowerPointLight Style**
**Output 14.41** PowerPointDark Style

![Output 14.41 PowerPointDark Style](image)

**Excel Style**

**Output 14.42** Excel Style

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Region</td>
<td>Division</td>
<td>Product type</td>
<td>Total Predicted Sales</td>
</tr>
<tr>
<td></td>
<td>EAST</td>
<td>CONSUMER</td>
<td>FURNITURE</td>
<td>$11,081</td>
</tr>
<tr>
<td>7</td>
<td>EDUCATION</td>
<td>FURNITURE</td>
<td>OFFICE</td>
<td>$21,939</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FURNITURE</td>
<td>OFFICE</td>
<td>$12,972</td>
</tr>
<tr>
<td></td>
<td></td>
<td>OFFICE</td>
<td></td>
<td>$16,434</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Division Total</td>
<td></td>
<td>$62,426</td>
</tr>
<tr>
<td></td>
<td>WEST</td>
<td>CONSUMER</td>
<td>FURNITURE</td>
<td>$10,286</td>
</tr>
<tr>
<td></td>
<td></td>
<td>OFFICE</td>
<td></td>
<td>$16,042</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FURNITURE</td>
<td>OFFICE</td>
<td>$12,816</td>
</tr>
<tr>
<td></td>
<td></td>
<td>OFFICE</td>
<td></td>
<td>$17,759</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Division Total</td>
<td></td>
<td>$56,903</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Grand Total</td>
<td></td>
<td>$119,329</td>
</tr>
</tbody>
</table>
General ODS Style Elements

The following table lists all the style elements available for ODS style definitions. The table provides a brief description of each style element and indicates the style elements from which it inherits its attributes. An abstract style element is one that is not used to generate any style element but provides a parent for one or more style elements to inherit.

<table>
<thead>
<tr>
<th>Style Element</th>
<th>Description</th>
<th>Inherits from</th>
</tr>
</thead>
<tbody>
<tr>
<td>Miscellaneous</td>
<td>Controls all container-oriented elements</td>
<td></td>
</tr>
<tr>
<td>Container *</td>
<td>Controls continued flag when a table breaks across a page (paginated destinations only)</td>
<td>TitlesAndFooters</td>
</tr>
<tr>
<td>Continued</td>
<td>Message when page won't fit (Printer only)</td>
<td>TitlesAndFooters</td>
</tr>
<tr>
<td>ExtendedPage</td>
<td>Controls page numbers for paginated destinations</td>
<td>TitlesAndFooters</td>
</tr>
<tr>
<td>PageNo</td>
<td>Controls space between tables in RTF output</td>
<td>TitlesAndFooters</td>
</tr>
<tr>
<td>Parskip</td>
<td>Controls the ODS RTF/MEASURED PREPAGE= style</td>
<td>TitlesAndFooters</td>
</tr>
</tbody>
</table>
### Table 15.2  Style Elements Affecting Documents

<table>
<thead>
<tr>
<th>Style Element</th>
<th>Description</th>
<th>Inherits from</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Documents</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Document</td>
<td>Controls the various document bodies. This generally includes things like the page background color and page margins.</td>
<td>Container *</td>
</tr>
<tr>
<td>Body</td>
<td>Controls the Body file</td>
<td>Document</td>
</tr>
<tr>
<td>Frame</td>
<td>Controls the Frame file for HTML</td>
<td>Document</td>
</tr>
<tr>
<td>Contents</td>
<td>Controls the Contents file</td>
<td>Document</td>
</tr>
<tr>
<td>Pages</td>
<td>Controls the Page file</td>
<td>Document</td>
</tr>
</tbody>
</table>

* An abstract style element. Abstract elements are not explicitly used in the ODS output. They are used for inheritance purposes only. Because of this, abstract styles will not appear in the output of destinations that generate a style sheet.
### Table 15.3  Style Elements Affecting Dates

<table>
<thead>
<tr>
<th>Style Element</th>
<th>Description</th>
<th>Inherits from</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dates</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BodyDate</td>
<td>Controls the date field in the Contents file</td>
<td>ContentsDate</td>
</tr>
<tr>
<td>Date</td>
<td>Controls how date fields look</td>
<td>Container*</td>
</tr>
<tr>
<td>PagesDate</td>
<td>Controls the date field in the Pages file</td>
<td>Date</td>
</tr>
</tbody>
</table>

* An abstract style element. Abstract elements are not explicitly used in the ODS output. They are used for inheritance purposes only. Because of this, abstract styles will not appear in the output of destinations that generate a style sheet.

### Table 15.4  Style Elements Affecting Table of Contents and Table of Pages

<table>
<thead>
<tr>
<th>Style Element</th>
<th>Description</th>
<th>Inherits from</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table of Contents and Table of Pages</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IndexItem</td>
<td>Controls list items and folders for Contents and Pages</td>
<td>Container*</td>
</tr>
<tr>
<td>ContentFolder</td>
<td>Controls the folders in the Contents file</td>
<td>IndexItem</td>
</tr>
<tr>
<td>ByContentFolder</td>
<td>Controls the byline folders in the Contents file</td>
<td>ContentFolder</td>
</tr>
<tr>
<td>ContentItem</td>
<td>Controls the items in the Contents file</td>
<td>IndexItem</td>
</tr>
<tr>
<td>PagesItem</td>
<td>Controls the items in the Pages file</td>
<td>IndexItem</td>
</tr>
<tr>
<td>Index</td>
<td>Controls miscellaneous Contents and Pages components</td>
<td>Container*</td>
</tr>
<tr>
<td>IndexProcName</td>
<td>Controls the PROC name in the Contents and Pages files</td>
<td>Index*</td>
</tr>
<tr>
<td>ContentProcName</td>
<td>Controls the PROC name in the Contents file</td>
<td>IndexProcName</td>
</tr>
<tr>
<td>ContentProcLabel</td>
<td>Controls the PROC label in the Contents file</td>
<td>ContentProcName</td>
</tr>
<tr>
<td>PagesProcName</td>
<td>Controls the PROC name in the Pages file</td>
<td>IndexProcName</td>
</tr>
</tbody>
</table>
## Style Elements

**Table of Contents and Table of Pages**

<table>
<thead>
<tr>
<th>Style Element</th>
<th>Description</th>
<th>Inherits from</th>
</tr>
</thead>
<tbody>
<tr>
<td>PagesProcLabel</td>
<td>Controls the PROC label in the Pages file</td>
<td>PagesProcName</td>
</tr>
<tr>
<td>IndexAction</td>
<td>Determines what happens on mouse-over events for folders and items (HTML only)</td>
<td>IndexItem</td>
</tr>
<tr>
<td>FolderAction</td>
<td>Determines what happens on mouse-over events for folders (HTML only)</td>
<td>IndexAction</td>
</tr>
<tr>
<td>IndexTitle</td>
<td>Controls the title of Contents and Pages files</td>
<td>Index *</td>
</tr>
<tr>
<td>ContentTitle</td>
<td>Controls the title of the Contents file.</td>
<td>IndexTitle</td>
</tr>
</tbody>
</table>

* An abstract style element. Abstract elements are not explicitly used in the ODS output. They are used for inheritance purposes only. Because of this, abstract styles will not appear in the output of destinations that generate a style sheet.

### Table 15.5 Style Elements Affecting Titles and Footers

<table>
<thead>
<tr>
<th>Style Element</th>
<th>Description</th>
<th>Inherits from</th>
</tr>
</thead>
<tbody>
<tr>
<td>System Titles and Footers</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SysTitleAndFooterContainer</td>
<td>Controls the container for system page title and system page footer. This element is usually used to add borders around a title.</td>
<td>Container</td>
</tr>
<tr>
<td>TitlesAndFooters</td>
<td>Controls system page title text and system page footer text</td>
<td>Container *</td>
</tr>
<tr>
<td>SystemTitle</td>
<td>Controls system title text</td>
<td>TitlesAndFooters</td>
</tr>
<tr>
<td>SystemTitle2</td>
<td>Controls system title2 text</td>
<td>SystemTitle</td>
</tr>
<tr>
<td>SystemTitle3</td>
<td>Controls system title3 text</td>
<td>SystemTitle2</td>
</tr>
<tr>
<td>SystemTitle4</td>
<td>Controls system title4 text</td>
<td>SystemTitle3</td>
</tr>
<tr>
<td>SystemTitle5</td>
<td>Controls system title5 text</td>
<td>SystemTitle4</td>
</tr>
<tr>
<td>SystemTitle6</td>
<td>Controls system title6 text</td>
<td>SystemTitle5</td>
</tr>
<tr>
<td>SystemTitle7</td>
<td>Controls system title7 text</td>
<td>SystemTitle6</td>
</tr>
</tbody>
</table>
Table 15.6  Style Elements Affecting Procedure Titles

<table>
<thead>
<tr>
<th>Style Element</th>
<th>Description</th>
<th>Inherits from</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROC Titles</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TitleAndNoteContainer</td>
<td>Controls the container for procedure-defined titles and notes</td>
<td>Container</td>
</tr>
<tr>
<td>ProcTitle</td>
<td>Controls procedure title text</td>
<td>TitlesAndFooters</td>
</tr>
<tr>
<td>ProcTitleFixed</td>
<td>Controls procedure title text that requests a fixed font</td>
<td>ProcTitle</td>
</tr>
</tbody>
</table>
### Table 15.7  Style Elements Affecting Bylines

<table>
<thead>
<tr>
<th>Style Element</th>
<th>Description</th>
<th>Inherits from</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bylines</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BylineContainer</td>
<td>Controls the container for the byline. This is generally used to add borders to a byline.</td>
<td>Container</td>
</tr>
<tr>
<td>Byline</td>
<td>Controls byline text</td>
<td>TitlesAndFooters</td>
</tr>
</tbody>
</table>

### Table 15.8  Style Elements Affecting Notes, Warnings, and Errors

Notes, warnings, and errors consist of two pieces: a banner area and a content area as shown in the following diagram. The Banner elements generally print the content of the banner (that is, "NOTE: ", "WARNING: ", and so on) using the PRETEXT= attribute.

<table>
<thead>
<tr>
<th>Style Element</th>
<th>Description</th>
<th>Inherits from</th>
</tr>
</thead>
<tbody>
<tr>
<td>Note</td>
<td>Controls the container for note banners and note contents</td>
<td>Container*</td>
</tr>
<tr>
<td>NoteBanner</td>
<td>Controls the banner for NOTE:s</td>
<td>Note</td>
</tr>
<tr>
<td>NoteContent</td>
<td>Controls the contents for NOTE:s</td>
<td>Note</td>
</tr>
<tr>
<td>NoteContentFixed</td>
<td>Controls the contents for NOTE:s. Fixed font.</td>
<td>NoteContent</td>
</tr>
<tr>
<td>WarnBanner</td>
<td>Controls the banner for WARNING:s</td>
<td>Note</td>
</tr>
<tr>
<td>WarnContent</td>
<td>Controls the contents of WARNING:s</td>
<td>Note</td>
</tr>
<tr>
<td>WarnContentFixed</td>
<td>Controls the contents for WARNING:s. Fixed font.</td>
<td>WarnContent</td>
</tr>
<tr>
<td>ErrorBanner</td>
<td>Controls the banner for ERROR:s</td>
<td>Note</td>
</tr>
<tr>
<td>ErrorContent</td>
<td>Controls the contents of ERROR:s</td>
<td>Note</td>
</tr>
<tr>
<td>ErrorContentFixed</td>
<td>Controls the contents for ERROR:s. Fixed font.</td>
<td>ErrorContent</td>
</tr>
<tr>
<td>Style Element</td>
<td>Description</td>
<td>Inherits from</td>
</tr>
<tr>
<td>--------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>----------------</td>
</tr>
<tr>
<td>Notes, Warnings, and Errors</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FatalBanner</td>
<td>Controls the banner for FATAL:s</td>
<td>Note</td>
</tr>
<tr>
<td>FatalContent</td>
<td>Controls the contents of FATAL:s</td>
<td>Note</td>
</tr>
<tr>
<td>FatalContentFixed</td>
<td>Controls the contents for FATAL:s. Fixed font.</td>
<td>FatalContent</td>
</tr>
</tbody>
</table>

* An abstract style element. Abstract elements are not explicitly used in the ODS output. They are used for inheritance purposes only. Because of this, abstract styles will not appear in the output of destinations that generate a style sheet.

Table 15.9  Style Elements Affecting Tables and Batch Output

<table>
<thead>
<tr>
<th>Style Element</th>
<th>Description</th>
<th>Inherits from</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tables and Batch Output</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Output</td>
<td>Controls basic output forms. This is generally used to control the borders (using the FRAME=, RULES=, and individual border control attributes), cell spacing, cell padding, and background color.</td>
<td>Container*</td>
</tr>
<tr>
<td>Table</td>
<td>Controls overall table style</td>
<td>Output</td>
</tr>
<tr>
<td>Batch</td>
<td>Controls batch mode output</td>
<td>Output</td>
</tr>
<tr>
<td>TableHeaderContainer</td>
<td>Places and controls the box around all column headings (RTF only)</td>
<td>Container*</td>
</tr>
<tr>
<td>TableFooterContainer</td>
<td>Places and controls the box around all column footers (RTF only)</td>
<td>Container*</td>
</tr>
<tr>
<td>Style Element</td>
<td>Description</td>
<td>Inherits from</td>
</tr>
<tr>
<td>-------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>----------------</td>
</tr>
<tr>
<td><strong>Tables and Batch Output</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>ColumnGroup</strong></td>
<td>Places and controls the box around groups of columns (RTF only)</td>
<td>Container*</td>
</tr>
<tr>
<td><strong>Table Data Cells</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Cell</strong></td>
<td>Controls data, header, and footer cells</td>
<td>Container*</td>
</tr>
<tr>
<td><strong>Data</strong></td>
<td>Default style for data cells</td>
<td>Cell</td>
</tr>
<tr>
<td><strong>DataFixed</strong></td>
<td>Default style for data cells that request a fixed font</td>
<td>Data</td>
</tr>
<tr>
<td><strong>DataEmpty</strong></td>
<td>Controls emphasized data cells</td>
<td>Data</td>
</tr>
<tr>
<td><strong>DataEmphasis</strong></td>
<td>Controls emphasized data cells</td>
<td>Data</td>
</tr>
<tr>
<td><strong>DataEmphasisFixed</strong></td>
<td>Controls emphasized data cells that request a fixed font</td>
<td>DataEmphasis</td>
</tr>
<tr>
<td>Style Element</td>
<td>Description</td>
<td>Inherits from</td>
</tr>
<tr>
<td>--------------------</td>
<td>--------------------------------------------------</td>
<td>-----------------</td>
</tr>
<tr>
<td><strong>Data Strong</strong></td>
<td>Controls strong (more emphasized) data cells</td>
<td>Data</td>
</tr>
<tr>
<td><strong>Data Strong Fixed</strong></td>
<td>Controls strong (more emphasized) data cells that request a fixed font</td>
<td>Data Strong</td>
</tr>
</tbody>
</table>

* An abstract style element. Abstract elements are not explicitly used in the ODS output. They are used for inheritance purposes only. Because of this, abstract styles will not appear in the output of destinations that generate a style sheet.

Table 15.11  **Style Elements Affecting Header and Footer Cells**

<table>
<thead>
<tr>
<th>Style Element</th>
<th>Description</th>
<th>Inherits from</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Headers And Footers</strong></td>
<td>Controls table headers and footers</td>
<td>Cell*</td>
</tr>
<tr>
<td><strong>Header</strong></td>
<td>Controls the headers of a table</td>
<td>Headers And Footers</td>
</tr>
<tr>
<td><strong>Header Fixed</strong></td>
<td>Controls the header of a table that request a fixed font</td>
<td>Header</td>
</tr>
<tr>
<td><strong>Header Empty</strong></td>
<td>Controls empty table header cells</td>
<td>Header</td>
</tr>
<tr>
<td><strong>Header Emphasis</strong></td>
<td>Controls emphasized table header cells</td>
<td>Header</td>
</tr>
<tr>
<td><strong>Header Emphasis Fixed</strong></td>
<td>Controls emphasized table header cells that request a fixed font</td>
<td>Header Emphasis</td>
</tr>
<tr>
<td><strong>Header Strong</strong></td>
<td>Controls strong (more emphasized) table header cells</td>
<td>Header</td>
</tr>
<tr>
<td><strong>Header Strong Fixed</strong></td>
<td>Controls strong (more emphasized) table header cells</td>
<td>Header Strong</td>
</tr>
<tr>
<td><strong>Row Header</strong></td>
<td>Controls row headers</td>
<td>Header</td>
</tr>
<tr>
<td><strong>Row Header Fixed</strong></td>
<td>Controls row headers that request a fixed font</td>
<td>Row Header</td>
</tr>
<tr>
<td><strong>Row Header Empty</strong></td>
<td>Controls empty row headers</td>
<td>Row Header</td>
</tr>
<tr>
<td><strong>Row Header Emphasis</strong></td>
<td>Controls emphasized row headers</td>
<td>Row Header</td>
</tr>
<tr>
<td>Style Element</td>
<td>Description</td>
<td>Inherits from</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>-------------------------------------------------------</td>
<td>---------------------</td>
</tr>
<tr>
<td><strong>Table Header and Footer Cells</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RowHeaderEmphasisFixed</td>
<td>Controls emphasized row headers that request a fixed font</td>
<td>RowHeaderEmphasis</td>
</tr>
<tr>
<td>RowHeaderStrong</td>
<td>Controls strong (more emphasized) row headers</td>
<td>RowHeader</td>
</tr>
<tr>
<td>RowHeaderStrongFixed</td>
<td>Controls strong (more emphasized) row headers that request a fixed font</td>
<td>RowHeaderStrong</td>
</tr>
<tr>
<td>Footer</td>
<td>Controls table footers</td>
<td>HeadersAndFooters</td>
</tr>
<tr>
<td>FooterFixed</td>
<td>Controls table footers that request a fixed font</td>
<td>Footer</td>
</tr>
<tr>
<td>FooterEmpty</td>
<td>Controls empty table footers</td>
<td>Footer</td>
</tr>
<tr>
<td>FooterEmphasis</td>
<td>Controls emphasized table footers</td>
<td>Footer</td>
</tr>
<tr>
<td>FooterEmphasisFixed</td>
<td>Controls emphasized table footers that request a fixed font</td>
<td>FooterEmphasis</td>
</tr>
<tr>
<td>FooterStrong</td>
<td>Controls strong (more emphasized) table footers</td>
<td>Footer</td>
</tr>
<tr>
<td>FooterStrongFixed</td>
<td>Controls strong (more emphasized) table footers that request a fixed font</td>
<td>FooterStrong</td>
</tr>
<tr>
<td>RowFooter</td>
<td>Controls a row footer (label)</td>
<td>Footer</td>
</tr>
<tr>
<td>RowFooterFixed</td>
<td>Controls a row footer (label) that request a fixed font</td>
<td>RowFooter</td>
</tr>
<tr>
<td>RowFooterEmpty</td>
<td>Controls an empty row footer (label)</td>
<td>RowFooter</td>
</tr>
<tr>
<td>RowFooterEmphasis</td>
<td>Controls an emphasized row footer (label)</td>
<td>RowFooter</td>
</tr>
<tr>
<td>RowFooterEmphasisFixed</td>
<td>Controls an emphasized row footer (label) that request a fixed font</td>
<td>RowFooterEmphasis</td>
</tr>
<tr>
<td>RowFooterStrong</td>
<td>Controls a strong (more emphasized) row footer (label)</td>
<td>RowFooter</td>
</tr>
</tbody>
</table>
**Table 15.12 Style Elements Affecting PROC TABULATE Captions**

<table>
<thead>
<tr>
<th>Style Element</th>
<th>Description</th>
<th>Inherits from</th>
</tr>
</thead>
<tbody>
<tr>
<td>Caption</td>
<td>Controls captions in PROC TABULATE</td>
<td>HeadersAndFooters*</td>
</tr>
<tr>
<td>BeforeCaption</td>
<td>Controls the caption that comes before a table</td>
<td>Caption</td>
</tr>
<tr>
<td>AfterCaption</td>
<td>Controls the caption that comes after a table</td>
<td>Caption</td>
</tr>
</tbody>
</table>

* An abstract style element. Abstract elements are not explicitly used in the ODS output. They are used for inheritance purposes only. Because of this, abstract styles will not appear in the output of destinations that generate a style sheet.

## Style Elements Affecting Template-Based Graphics

The following style elements affect template-based graphics and can be specified by Graph Template Language appearance options or used in styles. Template-based graphics include all SAS/GRAPH output where a compiled ODS template of type STATGRAPH is used to produce graphical output. Supplied templates are stored in Sashelp.Tmplmst. Device drivers and some global statements such as SYMBOL, PATTERN, AXIS, and LEGEND have no affect on this form of graphics. Common SAS/GRAPH procedures that produce template-based graphics are SGPLOT, SGPANEL, SGSCATTER, and SGRENDER in addition to many SAS/STAT, SAS/ETS, and SAS/QC procedures. ODS graphics always produce output as image files and use the ODS GRAPHICS statement to control the graphical environment.

Certain style elements were created to be used with specific plots or graphs. For example, the style element GraphFit2 is best used to modify secondary fit lines. The style element GraphConfidence2 was created to modify secondary confidence bands. The table below lists each style element, the portion of the graph that it affects or was created to use with, and the default attribute values. Attribute values can be changed with PROC TEMPLATE, as stated above.
For complete documentation on the style attributes that can be specified in each style element, see “Style Attributes Overview” in *SAS Output Delivery System: Procedures Guide*.

**Table 15.13  Graph Style Elements: General Graph Appearance**

<table>
<thead>
<tr>
<th>Style Element</th>
<th>Portion of Graph Affected</th>
<th>Recognized Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Graph</td>
<td>Graph size and outer border appearance</td>
<td>OutputWidth</td>
</tr>
<tr>
<td></td>
<td></td>
<td>OutputHeight</td>
</tr>
<tr>
<td></td>
<td></td>
<td>BorderColor</td>
</tr>
<tr>
<td></td>
<td></td>
<td>BorderWidth</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CellPadding</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CellSpacing</td>
</tr>
<tr>
<td>GraphAnnoLine</td>
<td>Annotation lines</td>
<td>ContrastColor</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LineStyle</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LineThickness</td>
</tr>
<tr>
<td>GraphAnnoShape</td>
<td>Annotation closed shapes such as circles, and squares</td>
<td>Color</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ContrastColor</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LineThickness</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LineStyle</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Transparency</td>
</tr>
<tr>
<td>GraphAnnoText</td>
<td>Annotation text</td>
<td>Font</td>
</tr>
<tr>
<td></td>
<td></td>
<td>* or font-attributes*</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Color</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MarkerSize</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MarkerSymbol</td>
</tr>
<tr>
<td>GraphAxisLines</td>
<td>X, Y and Z axis lines</td>
<td>ContrastColor</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LineStyle</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LineThickness</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TickDisplay</td>
</tr>
<tr>
<td>GraphBackgroundColor</td>
<td>Background of the graph</td>
<td>Color</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Transparency</td>
</tr>
<tr>
<td>GraphBorderLines</td>
<td>Border around graph wall, legend border, borders to complete axis frame</td>
<td>ContrastColor</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LineThickness</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LineStyle</td>
</tr>
<tr>
<td>GraphDataText</td>
<td>Text font and color for point and line labels</td>
<td>Font</td>
</tr>
<tr>
<td></td>
<td></td>
<td>* or font-attributes*</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Color</td>
</tr>
<tr>
<td>Style Element</td>
<td>Portion of Graph Affected</td>
<td>Recognized Attributes</td>
</tr>
<tr>
<td>----------------------------</td>
<td>----------------------------------------------------------------------------------------</td>
<td>----------------------------------------</td>
</tr>
<tr>
<td>GraphFootnoteText</td>
<td>Text font and color for footnote(s)</td>
<td>Font</td>
</tr>
<tr>
<td></td>
<td></td>
<td>or font-attributes*</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Color</td>
</tr>
<tr>
<td>GraphGridLines</td>
<td>Horizontal and vertical grid lines drawn at major tick marks</td>
<td>Color</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ContrastColor</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DisplayOpts</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LineStyle</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LineThickness</td>
</tr>
<tr>
<td>GraphHeaderBackground</td>
<td>Background color of the legend title</td>
<td>Color</td>
</tr>
<tr>
<td>GraphLabelText</td>
<td>Text font and color for axis labels and legend titles</td>
<td>Font</td>
</tr>
<tr>
<td></td>
<td></td>
<td>or font-attributes*</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Color</td>
</tr>
<tr>
<td>GraphLegendBackground</td>
<td>Background color of the legend</td>
<td>Color</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FrameBorder</td>
</tr>
<tr>
<td>GraphMinorGridLines</td>
<td>Appearance of the grid lines.</td>
<td>ContrastColor</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DisplayOpts</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LineStyle</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LineThickness</td>
</tr>
<tr>
<td>GraphOutlines</td>
<td>Outline properties for fill areas such as bars, pie slices, box plots, ellipses, and histograms</td>
<td>Color</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ContrastColor</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LineStyle</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LineThickness</td>
</tr>
<tr>
<td>GraphReference</td>
<td>Horizontal and vertical reference lines and drop lines</td>
<td>ContrastColor</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LineStyle</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LineThickness</td>
</tr>
<tr>
<td>GraphTitleText</td>
<td>Text font and color for title(s)</td>
<td>Font</td>
</tr>
<tr>
<td></td>
<td></td>
<td>or font-attributes*</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Color</td>
</tr>
<tr>
<td>GraphUnicodeText</td>
<td>Text font for Unicode values</td>
<td>Font</td>
</tr>
<tr>
<td></td>
<td></td>
<td>or font-attributes*</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Color</td>
</tr>
<tr>
<td>GraphValueText</td>
<td>Text font and color for axis tick values and legend values</td>
<td>Font</td>
</tr>
<tr>
<td></td>
<td></td>
<td>or font-attributes*</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Color</td>
</tr>
<tr>
<td>Style Element</td>
<td>Portion of Graph Affected</td>
<td>Recognized Attributes</td>
</tr>
<tr>
<td>--------------------</td>
<td>--------------------------------------------</td>
<td>------------------------------------------------------------</td>
</tr>
<tr>
<td>GraphWalls</td>
<td>Vertical wall(s) bounded by axes</td>
<td>Color&lt;br&gt;FrameBorder&lt;br&gt;LineThickness&lt;br&gt;LineStyle&lt;br&gt;ContrastColor</td>
</tr>
<tr>
<td></td>
<td></td>
<td><em>Font-attributes</em> can be one of the following: FONTFAMILY=, FONTSIZE=, FONTSTYLE=, FONTWEIGHT=.*</td>
</tr>
</tbody>
</table>

*Table 15.14 Style Elements Affecting Graphical Data Representation*

<table>
<thead>
<tr>
<th>Style Element</th>
<th>Portion of Graph Affected</th>
<th>Recognized Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>GraphBoxMean</td>
<td>Marker for mean</td>
<td>ContrastColor&lt;br&gt;MarkerSize&lt;br&gt;MarkerSymbol</td>
</tr>
<tr>
<td>GraphBoxMedian</td>
<td>Line for median</td>
<td>ContrastColor&lt;br&gt;LineStyle&lt;br&gt;LineThickness</td>
</tr>
<tr>
<td>GraphBoxWhisker</td>
<td>Box whiskers and serifs</td>
<td>ContrastColor&lt;br&gt;LineStyle&lt;br&gt;LineThickness</td>
</tr>
<tr>
<td>GraphConfidence</td>
<td>Primary confidence lines and bands, colors for bands and lines</td>
<td>ContrastColor&lt;br&gt;Color&lt;br&gt;MarkerSize&lt;br&gt;MarkerSymbol&lt;br&gt;LineStyle&lt;br&gt;LineThickness</td>
</tr>
<tr>
<td>GraphConfidence2</td>
<td>Secondary confidence lines and bands, color for bands, and contrast color for lines</td>
<td>ContrastColor&lt;br&gt;Color&lt;br&gt;MarkerSize&lt;br&gt;MarkerSymbol&lt;br&gt;LineStyle&lt;br&gt;LineThickness</td>
</tr>
<tr>
<td>GraphConnectLine</td>
<td>Line for connecting boxes or bars</td>
<td>ContrastColor&lt;br&gt;LineStyle&lt;br&gt;LineThickness</td>
</tr>
<tr>
<td>Style Element</td>
<td>Portion of Graph Affected</td>
<td>Recognized Attributes</td>
</tr>
<tr>
<td>------------------------</td>
<td>--------------------------------------------</td>
<td>----------------------------------------</td>
</tr>
<tr>
<td>GraphDataDefault</td>
<td>Primitives related to non-grouped data items, colors for filled areas, markers, and lines</td>
<td>ContrastColor, Color, MarkerSize, MarkerSymbol, LineStyle, LineThickness, StartColor, NeutralColor, EndColor</td>
</tr>
<tr>
<td>GraphCutLine</td>
<td>Cutline attributes for a dendogram</td>
<td>Color, LineStyle</td>
</tr>
<tr>
<td>GraphDataDefault</td>
<td>Primitives related to non-grouped data items, colors for filled areas, markers, and lines</td>
<td>Color, ContrastColor, MarkerSymbol, MarkerSize, LineStyle, LineThickness, StartColor, NeutralColor, EndColor</td>
</tr>
<tr>
<td>GraphError</td>
<td>Error line or error bar fill, ContrastColor for lines, Color for bar fill</td>
<td>CapStyle, ContrastColor, Color, LineStyle, Transparency</td>
</tr>
<tr>
<td>GraphFit</td>
<td>Primary fit lines such as a normal density curve</td>
<td>ContrastColor, Color, MarkerSize, MarkerSymbol, LineStyle, LineThickness</td>
</tr>
<tr>
<td>GraphFit2</td>
<td>Secondary fit lines such as a kernel density curve</td>
<td>ContrastColor, Color, MarkerSize, MarkerSymbol, LineStyle, LineThickness</td>
</tr>
<tr>
<td>Style Element</td>
<td>Portion of Graph Affected</td>
<td>Recognized Attributes</td>
</tr>
<tr>
<td>-------------------</td>
<td>----------------------------------------------------------------</td>
<td>----------------------------------------</td>
</tr>
<tr>
<td>GraphFinal</td>
<td>Final data for the waterfall chart. Color applies to filled areas.</td>
<td>Color, ContrastColor, LineStyle, LineThickness, MarkerSize, MarkerSymbol</td>
</tr>
<tr>
<td>GraphInitial</td>
<td>Initial data for the waterfall chart. Color applies to filled areas.</td>
<td>Color, ContrastColor, LineStyle, LineThickness, MarkerSize, MarkerSymbol</td>
</tr>
<tr>
<td>GraphMissing</td>
<td>Properties for graph items representing missing values</td>
<td>ContrastColor, Color, MarkerSymbol, MarkerSize, LineStyle, LineThickness, Transparency</td>
</tr>
<tr>
<td>GraphOther</td>
<td>Other data for the graph. Color applies to filled areas.</td>
<td>Color, ContrastColor, LineStyle, LineThickness, MarkerSize, MarkerSymbol</td>
</tr>
<tr>
<td>GraphOverflow</td>
<td>Overflow data for the graph. Color applies to filled areas. ContrastColor applies to markers and lines.</td>
<td>Color, ContrastColor, LineStyle, LineThickness, MarkerSize, MarkerSymbol</td>
</tr>
<tr>
<td>GraphOutlier</td>
<td>Outlier data for the graph</td>
<td>ContrastColor, Color, MarkerSize, MarkerSymbol</td>
</tr>
<tr>
<td><strong>Style Element</strong></td>
<td><strong>Portion of Graph Affected</strong></td>
<td><strong>Recognized Attributes</strong></td>
</tr>
<tr>
<td>-----------------------------</td>
<td>---------------------------------------------------------------------------------------------</td>
<td>------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>GraphPrediction</td>
<td>Prediction lines</td>
<td>ContrastColor, Color,LineStyle, LineThickness, MarkerSize, MarkerSymbol</td>
</tr>
<tr>
<td>GraphPredictionLimits</td>
<td>Fills for prediction limits</td>
<td>ContrastColor, Color, MarkerSize, MarkerSymbol</td>
</tr>
<tr>
<td>GraphUnderflow</td>
<td>Underflow data for the graph. Color applies to filled areas. ContrastColor applies to markers and lines.</td>
<td>Color, ContrastColor, LineStyle, LineThickness, MarkerSize, MarkerSymbol, TextColor</td>
</tr>
<tr>
<td>GraphSelection</td>
<td>For interactive graphs, visual properties of selected item. Color for selected fill area, ContrastColor for selected marker or line.</td>
<td>ContrastColor, Color, MarkerSymbol, MarkerSize, LineStyle, LineThickness</td>
</tr>
<tr>
<td>ThreeColorAltRamp</td>
<td>Line contours, markers, and data labels with segmented range color response</td>
<td>StartColor, NeutralColor, EndColor</td>
</tr>
<tr>
<td>ThreeColorRamp</td>
<td>Gradient contours, surfaces, markers, and data labels with continuous color response</td>
<td>StartColor, NeutralColor, EndColor</td>
</tr>
<tr>
<td>TwoColorAltRamp</td>
<td>Line contours, markers, and data labels with segmented range color response</td>
<td>StartColor, EndColor</td>
</tr>
<tr>
<td>TwoColorRamp</td>
<td>Gradient contours, surfaces, markers, and data labels with continuous color response</td>
<td>StartColor, EndColor</td>
</tr>
</tbody>
</table>
## Table 15.15  Graphical Style Elements: Data Related (Grouped)

<table>
<thead>
<tr>
<th>Style Elements</th>
<th>Portion of Graph Affected</th>
<th>Recognized Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>GraphData1</td>
<td>Primitives related to the first 7 grouped data items. Color applies to filled areas.</td>
<td>Color</td>
</tr>
<tr>
<td></td>
<td>ContrastColor applies to markers and lines.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>FillPattern*</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LineStyle</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MarkerSymbol</td>
</tr>
<tr>
<td>GraphData2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GraphData3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GraphData4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GraphData5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GraphData6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GraphData7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GraphData8</td>
<td>Primitives related to the 8th through 11th grouped data items.</td>
<td>Color</td>
</tr>
<tr>
<td></td>
<td>ContrastColor</td>
<td></td>
</tr>
<tr>
<td></td>
<td>FillPattern*</td>
<td></td>
</tr>
<tr>
<td></td>
<td>LineStyle</td>
<td></td>
</tr>
<tr>
<td>GraphData9</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GraphData10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GraphData11</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GraphData12</td>
<td>Primitives related to the 12th grouped data item.</td>
<td>Color</td>
</tr>
<tr>
<td></td>
<td>ContrastColor</td>
<td></td>
</tr>
<tr>
<td></td>
<td>FillPattern*</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GraphData13</td>
<td></td>
<td>FillPattern</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GraphData14</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GraphData15</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

* Style attribute FillPattern and style elements GraphData13–GraphData15 are available only with the JOURNAL2, JOURNAL3, and MONOCHROMEPRINTER styles.

## Table 15.16  Display Style Elements

<table>
<thead>
<tr>
<th>Style Element</th>
<th>Portion of Graph Affected</th>
<th>Recognized Attributes</th>
<th>Possible Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>GraphAltBlock</td>
<td>Alternate fill color for block plots</td>
<td>Color</td>
<td>GraphColors(&quot;gablock&quot;)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GraphBand</td>
<td>Display options for confidence bands</td>
<td>DisplayOpts</td>
<td>&quot;Fill&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GraphBar</td>
<td>Display options for bar charts</td>
<td>DisplayOpts</td>
<td>&quot;Fill outline&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GraphBox</td>
<td>Display options for box plots</td>
<td>DisplayOpts</td>
<td>&quot;Fill caps mean&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CapStyle</td>
<td>&quot;Median outliers&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Connect</td>
<td>&quot;Serif&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>&quot;Mean&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GraphBlock</td>
<td>Fill color for block plots</td>
<td>Color</td>
<td>GraphColors(&quot;gblock&quot;)</td>
</tr>
</tbody>
</table>
### Style Elements Affecting Device-Based Graphics

Device-based graphics are all SAS/GRAPH output where there is a user-specified or default device (DEVICE= option) that controls certain aspects of the graphical output. Supplied device drivers are stored in the Sashelp.Devices catalog. Examples of device drivers are SASPRTC, GIF, WIN, ACTIVEX, PDF, and SVG. Common SAS/GRAPH procedures that produce device-based graphics are GPLOT, GCHART, and GMAP. Most device-based graphics produce a GRSEG catalog entry as output and use the GOPTIONS statement to control the graphical environment.

For complete documentation on the style attributes that can be specified in each style element, see “Style Attributes Overview” in *SAS Output Delivery System: Procedures Guide*.

**Note:** These style elements affect device-based graphics only when the GSTYLE system option is in effect (this is the default for SAS 9.2). If the NOGSTYLE system option is specified, graphs do not use any style information. For more information about the GSTYLE system option, see *SAS System Options: Reference*.

### Table 15.17  Device-Based Graph Style Elements: General Graph Appearance

<table>
<thead>
<tr>
<th>Style Element</th>
<th>Portion of Graph Affected</th>
<th>Recognized Attributes</th>
<th>Possible Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>GraphEllipse</td>
<td>Display options for confidence ellipses</td>
<td>DisplayOpts</td>
<td>&quot;Outline&quot;</td>
</tr>
<tr>
<td>GraphHistogram</td>
<td>Display options for histograms</td>
<td>DisplayOpts</td>
<td>&quot;Fill outline&quot;</td>
</tr>
<tr>
<td>GraphSkins</td>
<td>One or more display features</td>
<td>DataSkin</td>
<td>CRISP</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>GLOSS</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>MATTE</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>NONE</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>PRESSED</td>
</tr>
<tr>
<td>KpiSkin</td>
<td></td>
<td>DataSkin</td>
<td>BASIC</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>MODERN</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>NONE</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>ONYX</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SATIN</td>
</tr>
</tbody>
</table>

*CRISP, GLOSS, MATTE, NONE, PRESSED, BASIC, MODERN, NONE, ONYX, SATIN*
<table>
<thead>
<tr>
<th>Style Element</th>
<th>Portion of Graph Affected</th>
<th>Recognized Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Graph</td>
<td>Graph size and outer border appearance</td>
<td>OutputWidth</td>
</tr>
<tr>
<td></td>
<td></td>
<td>OutputHeight</td>
</tr>
<tr>
<td></td>
<td></td>
<td>BorderColor</td>
</tr>
<tr>
<td></td>
<td></td>
<td>BorderWidth</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CellPadding</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CellSpacing</td>
</tr>
<tr>
<td>GraphAxisLines</td>
<td>X, Y, and Z axis lines</td>
<td>Color</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LineStyle</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LineThickness</td>
</tr>
<tr>
<td>GraphBackground</td>
<td>Background of the graph</td>
<td>Transparency</td>
</tr>
<tr>
<td></td>
<td></td>
<td>BackgroundColor</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Gradient_Direction</td>
</tr>
<tr>
<td></td>
<td></td>
<td>StartColor</td>
</tr>
<tr>
<td></td>
<td></td>
<td>EndColor</td>
</tr>
<tr>
<td></td>
<td></td>
<td>BackgroundImage</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Image</td>
</tr>
<tr>
<td></td>
<td></td>
<td>VerticalAlign</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TextAlign</td>
</tr>
<tr>
<td>GraphBorderLines</td>
<td>Border around graph wall, legend border, borders to complete axis frame</td>
<td>Color</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LineThickness</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LineStyle</td>
</tr>
<tr>
<td>GraphCharts</td>
<td>All charts within the graph</td>
<td>Transparency</td>
</tr>
<tr>
<td></td>
<td></td>
<td>BackgroundColor</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Gradient_Direction</td>
</tr>
<tr>
<td></td>
<td></td>
<td>StartColor</td>
</tr>
<tr>
<td></td>
<td></td>
<td>EndColor</td>
</tr>
<tr>
<td></td>
<td></td>
<td>BackgroundImage</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Image</td>
</tr>
<tr>
<td></td>
<td></td>
<td>VerticalAlign</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TextAlign</td>
</tr>
<tr>
<td>GraphDataText</td>
<td>Text font and color for point and line labels</td>
<td>Font or font-attributes*</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Color</td>
</tr>
<tr>
<td>Style Element</td>
<td>Portion of Graph Affected</td>
<td>Recognized Attributes</td>
</tr>
<tr>
<td>---------------------------</td>
<td>------------------------------------------------------------------------------------------</td>
<td>-----------------------</td>
</tr>
<tr>
<td>GraphFloor</td>
<td>3D floor</td>
<td>BackgroundColor</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Transparency</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Gradient_Direction</td>
</tr>
<tr>
<td></td>
<td></td>
<td>StartColor</td>
</tr>
<tr>
<td></td>
<td></td>
<td>EndColor</td>
</tr>
<tr>
<td></td>
<td></td>
<td>BackgroundImage</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Image</td>
</tr>
<tr>
<td></td>
<td></td>
<td>VerticalAlign</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TextAlign</td>
</tr>
<tr>
<td>GraphFootnoteText</td>
<td>Text font and color for footnotes</td>
<td>Font or font-attributes*</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Color</td>
</tr>
<tr>
<td>GraphGridLines</td>
<td>Horizontal and vertical grid lines drawn at major tick marks</td>
<td>Color</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LineStyle</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LineThickness</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Transparency</td>
</tr>
<tr>
<td></td>
<td></td>
<td>displayopts</td>
</tr>
<tr>
<td>GraphGridLines</td>
<td>Horizontal and vertical grid lines drawn at major tick marks</td>
<td>Color</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LineStyle</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LineThickness</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Transparency</td>
</tr>
<tr>
<td></td>
<td></td>
<td>displayopts</td>
</tr>
<tr>
<td>GraphLegendBackground</td>
<td>Background color of the legend</td>
<td>Color</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FrameBorder</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Transparency</td>
</tr>
<tr>
<td>GraphOutlines</td>
<td>Outline properties for fill areas such as bars, pie slices, and box plots.</td>
<td>Color</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LineStyle</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LineThickness</td>
</tr>
<tr>
<td>GraphTitle1Text</td>
<td>Text font and color for the first title</td>
<td>Font or font-attributes*</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Color</td>
</tr>
<tr>
<td>GraphTitleText</td>
<td>Text font and color for titles subsequent to the first title</td>
<td>Font or font-attributes*</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Color</td>
</tr>
<tr>
<td>GraphValueText</td>
<td>Text font and color for axis tick values and legend values</td>
<td>Font or font-attributes*</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Color</td>
</tr>
</tbody>
</table>
### Table 15.18  Style Elements Affecting Device-Based Non-Grouped Graphical Data Representation

<table>
<thead>
<tr>
<th>Style Element</th>
<th>Portion of Graph Affected</th>
<th>Recognized Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>GraphWalls</td>
<td>Vertical walls bounded by axes</td>
<td>Transparency, BackgroundColor, Gradient_Direction, StartColor, EndColor, BackgroundImage, Image</td>
</tr>
</tbody>
</table>

*Font-attributes can be one of the following: FONTFAMILY=, FONTSIZE=, FONTSTYLE=, FONTWEIGHT=.*
<table>
<thead>
<tr>
<th>Style Element</th>
<th>Portion of Graph Affected</th>
<th>Default Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>GraphOverflow</td>
<td>Overflow data for the graph. Color applies to filled areas. ContrastColor applies to markers and lines.</td>
<td>Color</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Color</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ContrastColor</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LineStyle</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LineThickness</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MarkerSize</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MarkerSymbol</td>
</tr>
<tr>
<td></td>
<td></td>
<td>textColor</td>
</tr>
<tr>
<td>GraphUnderflow</td>
<td>Underflow data for the graph. Color applies to filled areas. ContrastColor applies to markers and lines.</td>
<td>Color</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Color</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ContrastColor</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LineStyle</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LineThickness</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MarkerSize</td>
</tr>
<tr>
<td>ThreeColorAltRamp</td>
<td>Line contours, markers, and data labels with segmented range color response</td>
<td>StartColor</td>
</tr>
<tr>
<td>ThreeColorRamp</td>
<td>Gradient contours, surfaces, markers, and data labels with continuous color response</td>
<td>StartColor</td>
</tr>
<tr>
<td>TwoColorAltRamp</td>
<td>Line contours, markers, and data labels with segmented range color response</td>
<td>StartColor</td>
</tr>
<tr>
<td>TwoColorRamp</td>
<td>Gradient contours, surfaces, markers, and data labels with continuous color response</td>
<td>StartColor</td>
</tr>
</tbody>
</table>


<table>
<thead>
<tr>
<th>Style Element</th>
<th>Portion of Graph Affected</th>
<th>Default Attributes</th>
</tr>
</thead>
</table>
| GraphData1    | Primitives related to 1st grouped data items. Color applies to filled areas. ContrastColor applies to markers and lines. | BackGroundImage  
ContrastColor  
Color  
EndColor  
Gradient_Direction  
Image  
LineStyle  
LineThickness  
MarkerSize  
MarkerSymbol  
StartColor |
| GraphData2    | Primitives related to 2nd grouped data items | BackGroundImage  
ContrastColor  
Color  
EndColor  
Gradient_Direction  
Image  
LineStyle  
LineThickness  
MarkerSize  
MarkerSymbol  
StartColor |
| GraphData3    | Primitives related to 3rd grouped data items | BackGroundImage  
ContrastColor  
Color  
EndColor  
Gradient_Direction  
Image  
LineStyle  
LineThickness  
MarkerSize  
MarkerSymbol  
StartColor |
<table>
<thead>
<tr>
<th>Style Element</th>
<th>Portion of Graph Affected</th>
<th>Default Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>GraphData4</td>
<td>Primitives related to 4th grouped data items</td>
<td>BackGroundImage</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ContrastColor</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Color</td>
</tr>
<tr>
<td></td>
<td></td>
<td>EndColor</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Gradient_Direction</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Image</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LineStyle</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LineThickness</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MarkerSize</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MarkerSymbol</td>
</tr>
<tr>
<td></td>
<td></td>
<td>StartColor</td>
</tr>
<tr>
<td>GraphData5</td>
<td>Primitives related to 5th grouped data items</td>
<td>BackGroundImage</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ContrastColor</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Color</td>
</tr>
<tr>
<td></td>
<td></td>
<td>EndColor</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Gradient_Direction</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Image</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LineStyle</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LineThickness</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MarkerSize</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MarkerSymbol</td>
</tr>
<tr>
<td></td>
<td></td>
<td>StartColor</td>
</tr>
<tr>
<td>GraphData6</td>
<td>Primitives related to 6th grouped data items</td>
<td>BackGroundImage</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ContrastColor</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Color</td>
</tr>
<tr>
<td></td>
<td></td>
<td>EndColor</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Gradient_Direction</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Image</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LineStyle</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LineThickness</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MarkerSize</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MarkerSymbol</td>
</tr>
<tr>
<td></td>
<td></td>
<td>StartColor</td>
</tr>
<tr>
<td>Style Element</td>
<td>Portion of Graph Affected</td>
<td>Default Attributes</td>
</tr>
<tr>
<td>---------------</td>
<td>---------------------------</td>
<td>--------------------</td>
</tr>
<tr>
<td>GraphData7</td>
<td>Primitives related to 7th grouped data items</td>
<td>BackGroundImage, ContrastColor, Color, EndColor, Gradient_Direction, Image, LineStyle, LineThickness, MarkerSize, MarkerSymbol, StartColor</td>
</tr>
<tr>
<td>GraphData8</td>
<td>Primitives related to 8th grouped data items</td>
<td>BackGroundImage, ContrastColor, Color, EndColor, Gradient_Direction, Image, LineStyle, LineThickness, MarkerSize, MarkerSymbol, StartColor</td>
</tr>
<tr>
<td>GraphData9</td>
<td>Primitives related to 9th grouped data items</td>
<td>BackGroundImage, ContrastColor, Color, EndColor, Gradient_Direction, Image, LineStyle, LineThickness, MarkerSize, MarkerSymbol, StartColor</td>
</tr>
<tr>
<td>Style Element</td>
<td>Portion of Graph Affected</td>
<td>Default Attributes</td>
</tr>
<tr>
<td>-----------------</td>
<td>-----------------------------------------------</td>
<td>-------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>GraphData10</td>
<td>Primitives related to 10th grouped data items</td>
<td>BackGroundImage, ContrastColor, Color, EndColor, Gradient_Direction, Image, LineStyle, LineThickness, MarkerSize, MarkerSymbol, StartColor</td>
</tr>
<tr>
<td>GraphData11</td>
<td>Primitives related to 11th grouped data items</td>
<td>BackGroundImage, ContrastColor, Color, EndColor, Gradient_Direction, Image, LineStyle, LineThickness, MarkerSize, MarkerSymbol, StartColor</td>
</tr>
<tr>
<td>GraphData12</td>
<td>Primitives related to 12th grouped data items</td>
<td>BackGroundImage, ContrastColor, Color, EndColor, Gradient_Direction, Image, LineStyle, LineThickness, MarkerSize, MarkerSymbol, StartColor</td>
</tr>
</tbody>
</table>
Chapter 16

Style Attributes

Overview

Style attributes influence the characteristics of individual cells, tables, documents, graphs, and HTML frames. Style attributes exist within style elements and are specified by the `STYLE` statement or the `CLASS` statement. The default value for an attribute depends on the style that is in use. For information about styles, style elements, and style attributes, see “Understanding Styles, Style Elements, and Style Attributes” in SAS Output Delivery System: Procedures Guide. For information about using style attributes with ODS Statistical Graphics, see the chapter on controlling the appearance of your graphics in SAS Graph Template Language: User’s Guide.

Style attributes can be supplied by SAS or user-defined. Style attributes can be referenced with a style reference. See “Understanding Style References” in SAS Output Delivery System: Procedures Guide and “style-reference” on page 387 for more information.

The implementation of an attribute depends on the ODS destination that formats the output. When creating HTML output, the implementation of an attribute depends on the browser that is used. For information about viewing the attributes in a style, see “Viewing the Contents of a Style” in SAS Output Delivery System: Procedures Guide.

For a list of the values that style attributes can specify, see “Style Attribute Values” in SAS Output Delivery System: Procedures Guide. For a list of style elements that you can specify style attributes in, see Chapter 15, “Style Elements,” on page 311.

Style Attributes Tables

For detailed information about these style attributes and their aliases, see “Detailed Information for All Style Attributes” in SAS Output Delivery System: Procedures Guide.
### Table 16.1 Table of General Style Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Task</th>
<th>Destinations</th>
<th>Affected Items</th>
</tr>
</thead>
<tbody>
<tr>
<td>“ABSTRACT=ON</td>
<td>OFF” (p. 350)</td>
<td>Specify whether styles used in an HTML document are used in CSS style files</td>
<td>Markup family</td>
</tr>
<tr>
<td>“ACTIVELINKCOLOR=color” (p. 350)</td>
<td>Specify the color that a link in an HTML document changes to after you click it, but before the browser opens that file</td>
<td>Markup family</td>
<td>HTML documents</td>
</tr>
<tr>
<td>“ASIS=ON</td>
<td>OFF” (p. 350)</td>
<td>Specify how to handle leading spaces and line breaks in an HTML document</td>
<td>Markup family, printer family, and RTF</td>
</tr>
<tr>
<td>“BACKGROUNDCOLOR=color” (p. 351)</td>
<td>Specify the color of the background of tables, table cells, or graphs</td>
<td>Markup family, printer family, and RTF</td>
<td>Table cells, graphs</td>
</tr>
<tr>
<td>“BACKGROUNDIMAGE=string” (p. 351)</td>
<td>Specify an image to use as the background</td>
<td>Markup family, PCL, PS, and TAGSETS.RTF.</td>
<td>Table cells, graphs, graphs. For TAGSETS.RTF, only applies to the document.</td>
</tr>
<tr>
<td>“BACKGROUNDPOSITION=position” (p. 351)</td>
<td>Specify the position of the background of the tables, table cells, or graphs</td>
<td>Markup family, printer family, and RTF</td>
<td>Tables, graphs, and HTML documents</td>
</tr>
<tr>
<td>“BACKGROUNDREPEAT=option” (p. 352)</td>
<td>Specify whether an image is repeated horizontally, vertically, both, or not repeated</td>
<td>Markup family</td>
<td>Individual tables or table cells, graphs</td>
</tr>
<tr>
<td>“BODYSCROLLBAR=YES</td>
<td>NO</td>
<td>AUTO” (p. 352)</td>
<td>Specify whether to put a scroll bar in the frame that references the body file</td>
</tr>
<tr>
<td>“BODYSIZE=dimension</td>
<td>dimension%</td>
<td>* ” (p. 353)</td>
<td>Specify the width of the frame that displays the body file in the HTML frame file</td>
</tr>
<tr>
<td>“BORDERBOTTOMCOLOR=color” (p. 353)</td>
<td>Specify the color of the bottom border of the table</td>
<td>Markup family, printer family, RTF, and Measured RTF</td>
<td>Bottom border of a table or table cell</td>
</tr>
<tr>
<td>“BORDERBOTTOMSTYLE=line-style” (p. 353)</td>
<td>Specify the line style of the bottom border of the selected cell</td>
<td>Markup family, RTF, and Measured RTF</td>
<td>Bottom border of a table or table cell</td>
</tr>
<tr>
<td>Attribute</td>
<td>Task</td>
<td>Destinations</td>
<td>Affected Items</td>
</tr>
<tr>
<td>--------------------------------------------------</td>
<td>----------------------------------------------------------------------</td>
<td>---------------------------------------------------</td>
<td>-----------------------------------------------------</td>
</tr>
<tr>
<td>“BORDERBOTTOMWIDTH=dimension” (p. 354)</td>
<td>Specify the width of the bottom border of the table</td>
<td>Markup family, printer family, RTF, and Measured RTF</td>
<td>Bottom border of a table or table cell</td>
</tr>
<tr>
<td>“BORDERCOLLAPSE=COLLAPSE</td>
<td>SEPARATE” (p. 354)</td>
<td>Specify whether the border is collapsed or separated</td>
<td>Markup family, printer family, RTF, and Measured RTF</td>
</tr>
<tr>
<td>“BORDERCOLOR=color” (p. 354)</td>
<td>Specify the color of the border in a table or table cell if the border is just one color</td>
<td>Markup family, printer family, RTF, and Measured RTF</td>
<td>Individual tables or table cells</td>
</tr>
<tr>
<td>“BORDERCOLORDARK=color” (p. 354)</td>
<td>Specify the darker color to use in a border that uses two colors to create a three-dimensional effect</td>
<td>Markup family and printer family</td>
<td>Individual tables or table cells</td>
</tr>
<tr>
<td>“BORDERCOLORLIGHT=color” (p. 355)</td>
<td>Specify the lighter color to use in a border that uses two colors to create a three-dimensional effect</td>
<td>Markup family and printer family</td>
<td>Individual tables or table cells</td>
</tr>
<tr>
<td>“BORDERLEFTCOLOR=color” (p. 355)</td>
<td>Specify the color of the left border of a table</td>
<td>Markup family, printer family, RTF, and Measured RTF</td>
<td>Left border of a table or table cell</td>
</tr>
<tr>
<td>“BORDERLEFTSTYLE=line-style” (p. 355)</td>
<td>Specify the line style of the left border of the specified table cell</td>
<td>Markup family, RTF, and Measured RTF</td>
<td>Left border of the specified table cell</td>
</tr>
<tr>
<td>“BORDERLEFTWIDTH=dimension” (p. 355)</td>
<td>Specify the width of the left border of the table</td>
<td>Markup family, printer family, RTF, and Measured RTF</td>
<td>Left border of a table or table cell</td>
</tr>
<tr>
<td>“BORDERRIGHTCOLOR=color ” (p. 356)</td>
<td>Specify the color of the right border of the table</td>
<td>Markup family, printer family, RTF, and Measured RTF</td>
<td>Right border of a table or table cell</td>
</tr>
<tr>
<td>“BORDERRIGHTSTYLE=line-style” (p. 356)</td>
<td>Specify the line style of the right border of the selected cell</td>
<td>Markup family, RTF, and Measured RTF</td>
<td>Right border of the selected cell</td>
</tr>
<tr>
<td>“BORDERRIGHTWIDTH=dimension ” (p. 356)</td>
<td>Specify the width of the right border of the table</td>
<td>Markup family, printer family, RTF, and Measured RTF</td>
<td>Right border of a table</td>
</tr>
<tr>
<td>“BORDERSPACING=dimension” (p. 356)</td>
<td>Specify the thickness of the spacing between cells in a table</td>
<td>Markup family, RTF, and printer family</td>
<td>Tables</td>
</tr>
<tr>
<td>“BORDERTOPCOLOR=color” (p. 357)</td>
<td>Specify the color of the top border of the table</td>
<td>Markup family, printer family, RTF, and Measured RTF</td>
<td>Top border of a table or table cell</td>
</tr>
<tr>
<td>Attribute</td>
<td>Task</td>
<td>Destinations</td>
<td>Affected Items</td>
</tr>
<tr>
<td>-----------</td>
<td>------</td>
<td>--------------</td>
<td>----------------</td>
</tr>
<tr>
<td>“BORDERTOPSTYLE=\texttt{line-style}” (p. 357)</td>
<td>Specify the line style of the top border of the specified table cell</td>
<td>Markup family, RTF, and Measured RTF</td>
<td>Top border of the specified table cell</td>
</tr>
<tr>
<td>“BORDERTOPWIDTH=dimension” (p. 358)</td>
<td>Specify the width of the top border of the table</td>
<td>Markup family, printer family, RTF, and Measured RTF</td>
<td>Top border of a table</td>
</tr>
<tr>
<td>“BORDERWIDTH=dimension” (p. 358)</td>
<td>Specify the width of the border of the table</td>
<td>Markup family, RTF, and printer family</td>
<td>Individual tables or table cells</td>
</tr>
<tr>
<td>“CELLPADDING=dimension</td>
<td>dimension%” (p. 358)</td>
<td>Specify the amount of white space on each of the four sides of the content in a table cell</td>
<td>Markup family, RTF, and printer family</td>
</tr>
<tr>
<td>“CLASS='\texttt{string}’” (p. 359)</td>
<td>Specify the name of the style sheet class to use in an HTML document for the table or table cell</td>
<td>Markup family</td>
<td>Individual tables or table cells</td>
</tr>
<tr>
<td>“COLOR=color” (p. 359)</td>
<td>Specify the color of the foreground in tables, table cells, or graphs, which is primarily the color of text</td>
<td>Markup family, RTF, and printer family</td>
<td>Individual tables or table cells, graphs</td>
</tr>
<tr>
<td>“CONTENTPOSITION=\texttt{position}” (p. 360)</td>
<td>Specify the position, within the frame file, of the frames that display the contents and the page files</td>
<td>Markup family</td>
<td>Individual frames in HTML output</td>
</tr>
<tr>
<td>“CONTENTSCROLLBAR=YES</td>
<td>NO</td>
<td>AUTO” (p. 360)</td>
<td>Specify whether to put a scroll bar in the frames in the frame file that display the contents and the page files</td>
</tr>
<tr>
<td>“CONTENTS\texttt{SIZE}=\texttt{dimension}</td>
<td>dimension %</td>
<td>*” (p. 361)</td>
<td>Specify the width of the frames in the frame file that display the contents and the page files</td>
</tr>
<tr>
<td>“CONTENT\texttt{TYPE}='\texttt{string}’” (p. 361)</td>
<td>Specify the value of the content type for pages in an HTML document that is sent directly to a web server rather than to a file</td>
<td>Markup family</td>
<td>Individual frames in HTML output</td>
</tr>
<tr>
<td>“CONTRASTCOLOR=color” (p. 361)</td>
<td>Specify the alternate colors for maps</td>
<td>Markup family, RTF, and printer family</td>
<td>Graphs</td>
</tr>
<tr>
<td>Attribute</td>
<td>Task</td>
<td>Destinations</td>
<td>Affected Items</td>
</tr>
<tr>
<td>-----------</td>
<td>------</td>
<td>--------------</td>
<td>----------------</td>
</tr>
<tr>
<td>&quot;DOCTYPE=&quot;string&quot;&quot; (p. 363)</td>
<td>Specify the entire doctype declaration for the HTML document</td>
<td>Markup family</td>
<td>HTML documents</td>
</tr>
<tr>
<td>&quot;FILLRULEWIDTH=dimension&quot; (p. 364)</td>
<td>Place a rule of the specified width into the space around the text (or entire cell if there is no text) in a table where white space would otherwise appear</td>
<td>Printer family</td>
<td>HTML documents</td>
</tr>
<tr>
<td>&quot;FLYOVER=&quot;string&quot;&quot; (p. 364)</td>
<td>Specify the text to show in a data tip for the table cell</td>
<td>Markup family and PDF</td>
<td>Individual cells</td>
</tr>
<tr>
<td>&quot;FONT=font-definition&quot; (p. 364)</td>
<td>Specify a font definition to use in tables, table cells, and graphs</td>
<td>Markup family, RTF, and printer family</td>
<td>Individual tables or table cells, graphs</td>
</tr>
<tr>
<td>&quot;FONTFAMILY=&quot;string-1&lt;…, string-n&gt;&quot;&quot; (p. 365)</td>
<td>Specify the font to use in table cells and graphs</td>
<td>Markup family, RTF, and printer family</td>
<td>Individual tables or table cells, graphs</td>
</tr>
<tr>
<td>&quot;FONTSIZE=dimension</td>
<td>size&quot; (p. 365)</td>
<td>Specify the size of the font for tables, table cells, and graphs</td>
<td>Markup family, RTF, and printer family</td>
</tr>
<tr>
<td>&quot;FONTSTYLE=ITALIC</td>
<td>ROMAN</td>
<td>SLANT&quot; (p. 366)</td>
<td>Specify the style of the font for tables, table cells, and graphs</td>
</tr>
<tr>
<td>&quot;FONTWEIGHT=weight&quot; (p. 366)</td>
<td>Specify the font weight of tables, table cells, and graphs</td>
<td>Markup family, RTF, and printer family</td>
<td>Individual tables or table cells, graphs</td>
</tr>
<tr>
<td>&quot;FONTWIDTH=relative-width&quot; (p. 367)</td>
<td>Specify the font width of tables, table cells, and graphs compared to the width of the usual design of the table, table cell, or graph</td>
<td>Markup family, RTF, and printer family</td>
<td>Individual tables or table cells, graphs</td>
</tr>
<tr>
<td>&quot;FRAME=frame-type&quot; (p. 367)</td>
<td>Specify the type of frame to use on a table</td>
<td>Markup family, RTF, and printer family</td>
<td>Tables</td>
</tr>
<tr>
<td>&quot;FRAMEBORDER=ON</td>
<td>OFF&quot; (p. 368)</td>
<td>Specify whether to put a border around the frame for an HTML file that uses frames</td>
<td>Markup family</td>
</tr>
<tr>
<td>&quot;FRAMEBORDERWIDTH=dimension&quot; (p. 368)</td>
<td>Specify the width of the border around the frames for an HTML file that uses frames</td>
<td>Markup family</td>
<td>Individual frames in HTML output</td>
</tr>
<tr>
<td>Attribute</td>
<td>Task</td>
<td>Destinations</td>
<td>Affected Items</td>
</tr>
<tr>
<td>-----------</td>
<td>------</td>
<td>--------------</td>
<td>----------------</td>
</tr>
<tr>
<td>“FRAMESPACING=dimension” (p. 368)</td>
<td>Specify the width of the space between frames for HTML that uses frames</td>
<td>Markup family</td>
<td>Individual frames in HTML output</td>
</tr>
<tr>
<td>“HEIGHT=dimension” (p. 368)</td>
<td>Specify the height of a table cell, graph, or graphics in an HTML document ¹</td>
<td>Markup family, RTF, and printer family</td>
<td>Table cells, HTML documents, and graphs</td>
</tr>
<tr>
<td>“HREFTARGET=&quot;target&quot;” (p. 369)</td>
<td>Specify the window or frame in which to open the target of the link</td>
<td>Markup family</td>
<td>Individual cells</td>
</tr>
<tr>
<td>“HTMLID=&quot;string&quot;” (p. 370)</td>
<td>Specify an ID for the table or table cell</td>
<td>Markup family</td>
<td>Individual tables or table cells</td>
</tr>
<tr>
<td>“HTMLSTYLE=&quot;string&quot;” (p. 370)</td>
<td>Specify individual attributes and values for a table or table cell in an HTML document</td>
<td>Markup family</td>
<td>Individual tables or table cells</td>
</tr>
<tr>
<td>“IMAGE=&quot;string&quot;” (p. 370)</td>
<td>Specify the image to appear in a graph</td>
<td>Markup family, printer family, and RTF</td>
<td>Graphs</td>
</tr>
<tr>
<td>“LINKCOLOR=color” (p. 371)</td>
<td>Specify the color for the links in an HTML document that have not yet been visited</td>
<td>Markup family, printer family, and RTF</td>
<td>HTML documents</td>
</tr>
<tr>
<td>“LISTENTRYANCHOR=ON</td>
<td>OFF” (p. 371)</td>
<td>Specify whether to make the entry in the table of contents a link to the body file</td>
<td>Markup family</td>
</tr>
<tr>
<td>“LISTSTYLETYPE=bullet-type” (p. 371)</td>
<td>Specify the string to use for the bullets in the contents file</td>
<td>Markup family</td>
<td>Individual frames in HTML output</td>
</tr>
<tr>
<td>“MARGINBOTTOM=dimension” (p. 372)</td>
<td>Specify the bottom margin for the HTML document</td>
<td>Markup family, printer family, and RTF</td>
<td>HTML documents</td>
</tr>
<tr>
<td>“MARGINLEFT=dimension” (p. 372)</td>
<td>Specify the left margin for the HTML document</td>
<td>Markup family, printer family, and RTF</td>
<td>HTML documents</td>
</tr>
<tr>
<td>“MARGINRIGHT=dimension” (p. 373)</td>
<td>Specify the right margin for the HTML document</td>
<td>Markup family, printer family, and RTF</td>
<td>HTML documents</td>
</tr>
<tr>
<td>“MARGINTOP=dimension” (p. 373)</td>
<td>Specify the top margin for the HTML document</td>
<td>Markup family, printer family, and RTF</td>
<td>HTML documents</td>
</tr>
<tr>
<td>Attribute</td>
<td>Task</td>
<td>Destinations</td>
<td>Affected Items</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>-------------------------------------------</td>
<td>-------------------------------</td>
<td>----------------</td>
</tr>
<tr>
<td>“NOBREAKSPACE=ON</td>
<td>OFF ” (p. 374)</td>
<td>Specify how to handle space characters</td>
<td>Markup family, printer family, and RTF</td>
</tr>
<tr>
<td>“OVERHANGFACTOR=nonnegative-number” (p. 374)</td>
<td>Specify an upper limit for extending the width of the column in an HTML document</td>
<td>Markup family and printer family</td>
<td>HTML documents</td>
</tr>
<tr>
<td>“PADDING=dimension</td>
<td>dimension%” (p. 375)</td>
<td>Specify the amount of white space between the content of the table cell and the border</td>
<td>Markup family, RTF, and printer family</td>
</tr>
<tr>
<td>“PADDINGBOTTOM=dimension</td>
<td>dimension%” (p. 375)</td>
<td>Specify the amount of white space on the bottom of the content of the table cell</td>
<td>Markup family, RTF, and printer family</td>
</tr>
<tr>
<td>“PADDINGLEFT=dimension</td>
<td>dimension%” (p. 375)</td>
<td>Specify the amount of white space on the left side of the content of the table cell</td>
<td>Markup family, RTF, and printer family</td>
</tr>
<tr>
<td>“PADDINGRIGHT=dimension</td>
<td>dimension%” (p. 375)</td>
<td>Specify the amount of white space on the right side of the content of the table cell</td>
<td>Markup family, RTF, and printer family</td>
</tr>
<tr>
<td>“PADDINGTOP=dimension</td>
<td>dimension%” (p. 375)</td>
<td>Specify the amount of white space on the top of the content of the table cell</td>
<td>Markup family, RTF, and printer family</td>
</tr>
<tr>
<td>“PAGEBREAKHTML=&quot;string&quot;” (p. 375)</td>
<td>Specify HTML to place at page breaks in an HTML document</td>
<td>Markup family</td>
<td>HTML documents</td>
</tr>
<tr>
<td>“POSTHTML=&quot;string&quot;” (p. 376)</td>
<td>Specify the HTML code to place after the table or table cell</td>
<td>Markup family</td>
<td>Individual tables or table cells</td>
</tr>
<tr>
<td>“POSTIMAGE=&quot;string&quot;</td>
<td>fileref” (p. 376)</td>
<td>Specify an image to place before the table or table cell</td>
<td>Markup family</td>
</tr>
<tr>
<td>“POSTTEXT=&quot;string&quot;” (p. 376)</td>
<td>Specify text to place after the table cell or table</td>
<td>Markup family, printer family, and RTF</td>
<td>Individual tables or table cells</td>
</tr>
<tr>
<td>“PREHTML=&quot;string&quot;” (p. 376)</td>
<td>Specify the HTML code to place before the table or table cell</td>
<td>Markup family</td>
<td>Individual tables or table cells</td>
</tr>
<tr>
<td>Attribute</td>
<td>Task</td>
<td>Destinations</td>
<td>Affected Items</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>----------------------------------------------------------------------</td>
<td>--------------------------</td>
<td>------------------------------</td>
</tr>
<tr>
<td>“PREIMAGE=&quot;string&quot;</td>
<td>fileref” (p. 377)</td>
<td>Specify an image to place before the table or table cell</td>
<td>Markup family, printer family, and RTF</td>
</tr>
<tr>
<td>“PRETEXT=&quot;string”” (p. 377)</td>
<td>Specify text to place before the table cell or table</td>
<td>Markup family, printer family, and RTF</td>
<td>Individual tables or table cells</td>
</tr>
<tr>
<td>“PROTECTSPECIALCHARS=ON</td>
<td>OFF</td>
<td>AUTO” (p. 377)</td>
<td>Specify how less-than signs (&lt;), greater-than signs (&gt;), and ampersands (&amp;) are interpreted in table cells</td>
</tr>
<tr>
<td>“RULES=rule-type” (p. 378)</td>
<td>Specify the types of rules to use in tables</td>
<td>Markup family, printer family, and RTF</td>
<td>Tables</td>
</tr>
<tr>
<td>“STARTCOLOR=color” (p. 378)</td>
<td>Specify the start fill color for a graph</td>
<td>HTML</td>
<td>Graphs</td>
</tr>
<tr>
<td>“TAGATTR=&quot;string”” (p. 379)</td>
<td>Specify text to insert in the HTML</td>
<td>Markup family</td>
<td>Individual cells</td>
</tr>
<tr>
<td>“TEXTALIGN=alignment” (p. 379)</td>
<td>Specify justification in tables, table cells, and graphs</td>
<td>Printer family and RTF</td>
<td>Individual tables or table cells, graphs</td>
</tr>
<tr>
<td>“TEXTDECORATION=presentation–options” (p. 380)</td>
<td>Change the visual presentation of the text</td>
<td>Markup family, RTF, and printer family</td>
<td>Individual tables or table cells</td>
</tr>
<tr>
<td>“TEXTJUSTIFY=INTER_WORD</td>
<td>INTER_CHARACTER” (p. 381)</td>
<td>Specify if the words of the text are to be spaced evenly or if the characters are to be evenly justified</td>
<td>HTML, RTF, and TAGSETS.RTF</td>
</tr>
<tr>
<td>“TRANSPARENCY=dimension” (p. 381)</td>
<td>Specify a transparency level for graphs</td>
<td>HTML</td>
<td>Graphs</td>
</tr>
<tr>
<td>“URL=&quot;uniform-resource-locator”” (p. 381)</td>
<td>Specify a URL to link to</td>
<td>Markup family, RTF, and printer family</td>
<td>Individual cells</td>
</tr>
<tr>
<td>“VERTICALALIGN=BOTTOM</td>
<td>MIDDLE</td>
<td>TOP ” (p. 381)</td>
<td>Specify vertical justification</td>
</tr>
<tr>
<td>Attribute</td>
<td>Task</td>
<td>Destinations</td>
<td>Affected Items</td>
</tr>
<tr>
<td>-----------</td>
<td>------</td>
<td>--------------</td>
<td>----------------</td>
</tr>
<tr>
<td>“VISITEDLINKCOLOR=color” (p. 382)</td>
<td>Specify the color for links that have been visited in an HTML document</td>
<td>Markup family</td>
<td>HTML documents</td>
</tr>
<tr>
<td>“WATERMARK=ON</td>
<td>OFF” (p. 382)</td>
<td>Specify whether to make the image that is specified by BACKGROUNDIMAGE= into a &quot;watermark &quot;</td>
<td>Markup family</td>
</tr>
<tr>
<td>“WHITESPACE=options” (p. 382)</td>
<td>Specify how the browser handles multiple whitespace characters and line breaks.</td>
<td>Markup family, printer family, RTF, and Measured RTF</td>
<td>Lines of text</td>
</tr>
<tr>
<td>“WIDTH=dimension ” (p. 383)</td>
<td>Specify the width of a table cell, table, line, or a graph</td>
<td>Markup family, printer family, and RTF</td>
<td>Tables</td>
</tr>
</tbody>
</table>

1 This attribute can also be used to influence other characteristics as described in another section of the table.

*Note:* You can use the value _UNDEF_ for any style attribute. ODS treats an attribute that is set to _UNDEF_ as if its value had never been set, even in the parent or beyond.

Graphical style attributes can be used in graphical style elements for device-based graphics or template-based graphics (ODS graphics). Different style attributes are valid for different style elements. For a table of style elements and the style attributes that are valid in each one, see “Style Elements Affecting Template-Based Graphics” on page 321 and “Style Elements Affecting Device-Based Graphics” on page 329.

Device-based graphics are all SAS/GRAPH output where there is a user-specified or default device (DEVICE= option) that controls certain aspects of the graphical output. Supplied device drivers are stored in the SASHelp.Devices catalog. Examples of device drivers are SASPRTC, GIF, WIN, ACTIVEX, PDF, and SVG. Common SAS/GRAPH procedures that produce device-based graphics are GPOINT, GCHART, and GMAP. Most device-based graphics produce a GRSEG catalog entry as output and use the GOPTIONS statement to control the graphical environment.

Template-based graphics include all SAS/GRAPH output where a compiled ODS template of type STATGRAPH is used to produce graphical output. Supplied templates are stored in Sashelp.Tmplmst. Device drivers and some global statements such as SYMBOL, PATTERN, AXIS, and LEGEND have no affect on this form of graphics. Common SAS/GRAPH procedures that produce template-based graphics are SGPOINT, SG montage, and SGRENDER, in addition to many SAS/STAT, SAS/ETS, and SAS/QC procedures. ODS graphics always produce output as image files and use the ODS GRAPHICS statement to control the graphical environment.
**Table 16.2  Table of Graphical Style Attributes**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Task</th>
<th>Graphics Environment</th>
<th>Affected Items</th>
</tr>
</thead>
<tbody>
<tr>
<td>“BACKGROUNDIMAGE=&quot;string&quot;”  (p. 351)</td>
<td>Specify an image file path</td>
<td>Device-based graphics</td>
<td>Image that can be stretched, but not positioned in graph, chart, walls, floor</td>
</tr>
<tr>
<td>“CAPSTYLE=\texttt{line-shape}”  (p. 358)</td>
<td>Specify the shape of the line at the end of a box whisker graph</td>
<td>Template-based graphics</td>
<td>Shape of line at end of box whisker</td>
</tr>
<tr>
<td>“COLOR=\texttt{color}”  (p. 359)</td>
<td>Specify the color of the foreground in tables, table cells, or graphs, which is primarily the color of text</td>
<td>All graphics environments</td>
<td>Background color of the graph, walls, or floor; color of text</td>
</tr>
<tr>
<td>“CONNECT=\texttt{connect-line-type}”  (p. 359)</td>
<td>Specify characteristics of a box plot connect line</td>
<td>Template-based graphics</td>
<td>Box plot connect line</td>
</tr>
<tr>
<td>“CONTRASTCOLOR=\texttt{color}”  (p. 361)</td>
<td>Specify the color a line or marker</td>
<td>Template-based graphics</td>
<td>Color of line or marker</td>
</tr>
<tr>
<td>“DATASKIN=\texttt{CRISP</td>
<td>GLOSS</td>
<td>MATTE</td>
<td>NONE</td>
</tr>
<tr>
<td>“DISPLAYOPTS=&quot;display-feature&quot;”  (p. 362)</td>
<td>Specify display features for graphs</td>
<td>Template-based graphics</td>
<td>Displayed features of box plots, ellipses, histograms, bands</td>
</tr>
<tr>
<td>“DROP\texttt{SHADOW}=\texttt{ON</td>
<td>OFF} ”  (p. 363)</td>
<td>Specify whether the drop shadow color for text is displayed</td>
<td>Device-based graphics</td>
</tr>
<tr>
<td>“ENDCOLOR=\texttt{color}”  (p. 363)</td>
<td>Specify the final color used with a two- or three-color ramp</td>
<td>All graphics environments</td>
<td>Contours, gradient legends</td>
</tr>
<tr>
<td>“FILL\texttt{PATTERN}=\texttt{fillpattern-value}”  (p. 363)</td>
<td>Specify the fill pattern to be displayed on a chart</td>
<td>Template-based graphics</td>
<td></td>
</tr>
<tr>
<td>“FONT=\texttt{font-definition}”  (p. 364)</td>
<td>Specify a font definition to use in tables, table cells, and graphs</td>
<td>All graphics environments</td>
<td>All text font attributes</td>
</tr>
<tr>
<td>“FONTFAMILY=&quot;string-1&lt;…, string-n&gt;&quot;”  (p. 365)</td>
<td>Specify the font to use in table cells and graphs</td>
<td>All graphics environments</td>
<td>Font family</td>
</tr>
<tr>
<td>Attribute</td>
<td>Task</td>
<td>Graphics Environment</td>
<td>Affected Items</td>
</tr>
<tr>
<td>-----------</td>
<td>------</td>
<td>----------------------</td>
<td>----------------</td>
</tr>
<tr>
<td>“FONTSIZE=dimension</td>
<td>size” (p. 365)</td>
<td>Specify the size of the font for tables, table cells, and graphs</td>
<td>All graphics environments</td>
</tr>
<tr>
<td>“FONTSTYLE=ITALIC</td>
<td>ROMAN</td>
<td>SLANT” (p. 366)</td>
<td>Specify the style of the font for tables, table cells, and graphs</td>
</tr>
<tr>
<td>“FONTSTYLE=ITALIC</td>
<td>ROMAN</td>
<td>SLANT” (p. 366)</td>
<td>Specify the font weight of tables, table cells, and graphs</td>
</tr>
<tr>
<td>“FRAMEBORDER=ON</td>
<td>OFF” (p. 368)</td>
<td>Specify whether there is a graph wall border</td>
<td>All graphics environments</td>
</tr>
<tr>
<td>“GRADIENT_DIRECTION=</td>
<td>&quot;YAXIS&quot;</td>
<td>&quot;XAXIS&quot; “” (p. 368)</td>
<td>Specify the direction of the gradient</td>
</tr>
<tr>
<td>“IMAGE=&quot;string”” (p. 370)</td>
<td>Specify the path to an image</td>
<td>Device-based graphics</td>
<td>Image that can be positioned, but not stretched in graph, chart, walls, floor</td>
</tr>
<tr>
<td>“KPISKIN=BASIC</td>
<td>MODERN</td>
<td>NONE</td>
<td>ONYX</td>
</tr>
<tr>
<td>“LINESTYLE=pattern-number” (p. 370)</td>
<td>Specify the pattern of a line</td>
<td>All graphics environments</td>
<td>Borders, axis lines, grid, reference</td>
</tr>
<tr>
<td>“LINETHICKNESS=dimension” (p. 371)</td>
<td>Specify the thickness of a line</td>
<td>All graphics environments</td>
<td>Thickness of line</td>
</tr>
<tr>
<td>“MARKERSIZE=dimension” (p. 373)</td>
<td>Specify a marker size</td>
<td>All graphics environments</td>
<td>Marker size</td>
</tr>
<tr>
<td>“MARKERSYMBOL=marker-symbol” (p. 373)</td>
<td>Specify a marker symbol</td>
<td>All graphics environments</td>
<td>Marker used</td>
</tr>
<tr>
<td>“NEUTRALCOLOR=color” (p. 374)</td>
<td>Specify the middle color of a three-color ramp</td>
<td>Template-based graphics</td>
<td>Contours, gradient legends</td>
</tr>
<tr>
<td>“OUTPUTHEIGHT=dimension” (p. 374)</td>
<td>Specify the height of a graph</td>
<td>All graphics environments</td>
<td>Height of graph</td>
</tr>
<tr>
<td>Attribute</td>
<td>Task</td>
<td>Graphics Environment</td>
<td>Affected Items</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>------------------------------------------------</td>
<td>----------------------</td>
<td>-------------------------------------</td>
</tr>
<tr>
<td>“OUTPUTWIDTH=dimension” (p. 374)</td>
<td>Specify the width of a graph</td>
<td>All graphics environments</td>
<td>Width of graph</td>
</tr>
<tr>
<td>“STARTCOLOR=color” (p. 378)</td>
<td>Specify the start fill color for a graph</td>
<td>All graphics environments</td>
<td>Contours, gradient legends</td>
</tr>
<tr>
<td>“TEXTALIGN=alignment” (p. 379)</td>
<td>Specify the alignment of an image</td>
<td>Device-based graphics</td>
<td>Image horizontal positioning</td>
</tr>
<tr>
<td>“TICKDISPLAY=&quot;INSIDE&quot;</td>
<td>&quot;OUTSIDE&quot;</td>
<td>&quot;ACROSS&quot; ” (p. 381)</td>
<td>Specify the placement of all major and minor axis tick marks</td>
</tr>
<tr>
<td>“TRANSPARENCY=dimension” (p. 381)</td>
<td>Specify the transparency of backgrounds, fills, lines, and markers</td>
<td>All graphics environments</td>
<td>Backgrounds, fills, lines, markers</td>
</tr>
<tr>
<td>“VERTICALALIGN=BOTTOM</td>
<td>MIDDLE</td>
<td>TOP ” (p. 381)</td>
<td>Specify vertical justification</td>
</tr>
</tbody>
</table>

**Style Attributes Detailed Information**

**ABSTRACT=ON | OFF**
specifies whether styles used in an HTML document are used in CSS style files.

ON
specifies that styles are used in CSS style files.

OFF
specifies that styles are not used in CSS style files.

Restriction
The ABSTRACT= attribute is valid only in markup family destinations.

**ACTIVELINKCOLOR=color**
specifies the color that a link in an HTML document changes to after you click it, but before the browser opens that file.

Restriction
The ACTIVELINKCOLOR= attribute is valid only in markup family destinations.

See
color style attribute value on page 383

**ASIS=ON | OFF**
specifies how to handle leading spaces and line breaks in an HTML document.

ON
prints text with leading spaces and line breaks, in the same manner as the LISTING output.
### OFF
trims leading spaces and ignores line breaks.

**Default**
OFF

**Restriction**
The ASIS= attribute is valid only in markup family destinations, printer family destinations, and the RTF destination.

### BACKGROUNDCOLOR=\textit{color}
specifies the color of the background of the tables, table cells, or graphs.

**Alias**
BACKGROUND=

**Restriction**
The BACKGROUNDCOLOR= attribute is valid only in markup family destinations, printer family destinations, and the RTF destination.

**Interaction**
The CBACK= option in the SAS/GRAPH GOPTIONS statement overrides the BACKGROUNDCOLOR= attribute.

**Tip**
Generally, the background color of the table cell overrides the background color of the table. You see the background color for the table only as the space between table cells (see “BORDERSPACING=\textit{dimension}” on page 356).

**See**
color style attribute value on page 383

**Examples**
“Creating a Stand-Alone Style” in \textit{SAS Output Delivery System: Procedures Guide}

“Modifying the Default Style with the CLASS Statement” in \textit{SAS Output Delivery System: Procedures Guide}

### BACKGROUNDIMAGE=\texttt{"string"}
specifies an image in a table, table cell, or graph to use as the background. Viewers can tile or stretch the image as the background for the HTML table or graph that the procedure creates. For graphs, the specified image is stretched.

\texttt{string}

is the name of a GIF, JPEG, or PNG file. Use a simple filename, a complete path, or a URL. However, the most versatile approach is to use a simple filename and to place all image files in the local directory.

**Restriction**
The BACKGROUNDIMAGE= attribute is valid in markup family destinations, the PCL destination, and the PS destination. It is also valid in the TAGSETS.RTF destination, but the background image applies to the RTF document and not to a table or a table cell.

**Interaction**
The BACKGROUNDIMAGE= attribute is overridden by the IBACK= and IMAGESTYLE=FIT options in the SAS/GRAPH GOPTIONS statement.

**See**
string attribute value on page 388

### BACKGROUNDPOSITION=\textit{position}
specifies the position of the background of the tables, table cells, or graphs.
position can be one of the following:

- BOTTOM
- BOTTOM_CENTER
- BOTTOM_LEFT
- BOTTOM_RIGHT
- CENTER
- CENTER_BOTTOM
- CENTER_CENTER
- CENTER_LEFT
- CENTER_RIGHT
- CENTER_TOP
- LEFT
- LEFT_BOTTOM
- LEFT_CENTER
- LEFT_TOP
- RIGHT
- RIGHT_BOTTOM
- RIGHT_CENTER
- RIGHT_TOP
- TOP
- TOP_CENTER
- TOP_LEFT
- TOP_RIGHT

Default: TOP_LEFT

BACKGROUNDREPEAT=option
specifies whether an image is repeated horizontally, vertically, both, or not repeated. option can be one of the following:

- NO_REPEAT
  specifies that the image is not repeated.

- REPEAT
  specifies that the image is repeated both horizontally and vertically.

- REPEAT_X
  specifies that the image is repeated horizontally.

- REPEAT_Y
  specifies that the image is repeated vertically.

Restriction: The BACKGROUNDREPEAT= attribute is valid in most markup family destinations.

BODYSCROLLBAR=YES | NO | AUTO
specifies whether to put a scroll bar in the frame that references the body file.
YES
places a scroll bar in the frame that references the body file.

NO
specifies not to put a scroll bar in the frame that references the body file.

AUTO
places a scroll bar in the frame that references the body file only if needed.

Restriction
The BODYSIZE= attribute is valid only in markup family destinations.

Tip
Typically, BODYSIZE= is set to AUTO.

**BODYSIZE=**\textit{dimension} | \textit{dimension}\% | *
specifies the width of the frame that displays the body file in the HTML frame file.

\textit{dimension}
is a nonnegative number or the width of the frame specified as a percentage of the entire display.

*specifies to use whatever space is left after displaying the content and page files as specified by the CONTENTSIZE= attribute.

Restriction
The BODYSIZE= attribute is valid only in markup family destinations.

Tip
If \textit{dimension} is a nonnegative number, then the unit of measure is pixels.

See
dimension attribute value on page 385

For information about the HTML files that ODS creates, see “HTML Links and References Produced by the HTML Destination” in SAS Output Delivery System: User’s Guide.

**BORDERBOTTOMCOLOR=**\textit{color}
specifies the color of the bottom border of a table or table cell.

Restriction
The BORDERBOTTOMCOLOR= attribute is valid only in markup family destinations, printer family destinations, RTF destination, and the Measured RTF destination.

Tip
You might also need to specify a BORDERBOTTOMWIDTH= attribute to override the style in the ODS destination.

See
color style attribute value on page 383

**BORDERBOTTOMSTYLE=**\textit{line-style}
specifies the line style of the bottom border of the specified table cell.

\textit{line-style}
can be one of the following:

- DASHED
- DOTTED
- DOUBLE
- GROOVE
• HIDDEN
• INSET
• OUTSET
• RIDGE
• SOLID

Restriction: The BORDERBOTTOMSTYLE= attribute is valid only in markup family destinations, the RTF destination, and the Measured RTF destination.

Tip: You might also need to specify the BORDERBOTTOMWIDTH= attribute to override the style in the ODS destination.

**BORDERBOTTOMWIDTH=dimension**

specifies the width of the bottom border of a table or table cell.

Restriction: The BORDERBOTTOMWIDTH= attribute is valid only in markup family destinations, the RTF destination, printer family destinations, and the Measured RTF destination.

See: *dimension attribute value on page 385*

**BORDERCOLLAPSE=COLLAPSE | SEPARATE**

specifies whether the border is collapsed or separated.

Default: SEPARATE

**BORDERCOLOR=color**

specifies the border color of a table or table cell. The color is applied to all four borders.

Restriction: The BORDERCOLOR= attribute is valid only in markup family destinations, the RTF destination, printer family destinations, and the Measured RTF destination.

See: *color style attribute value on page 383*

**BORDERCOLORDARK=color**

in a table or table cell, specifies the darker color to use in a border that uses two colors to create a three-dimensional effect.

Restriction: The BORDERCOLORDARK= attribute is valid only in markup family destinations and printer family destinations.

Interaction: The BORDERCOLORDARK style attribute is ignored in HTML4 output because it is not part of the HTML4 standard. To create a color border in the HTML4 output, use the BORDERCOLOR= style attribute.

See: *color style attribute value on page 383*

Example: “Defining a Table and Graph Style” in *SAS Output Delivery System: Procedures Guide*
**BORDERCOLORLIGHT=** *color*

in a table or table cell, specifies the lighter color to use in a border that uses two colors to create a three-dimensional effect.

**Restriction**
The `BORDERCOLORLIGHT=` attribute is valid only in markup family destinations and printer family destinations.

**Interaction**
The `BORDERCOLORLIGHT` style attribute is ignored in the creation of HTML4 output because it is not part of the HTML4 standard. To create a color border in HTML4 output, use the `BORDERCOLOR=` style attribute.

**See**
color style attribute value on page 383

**Example**
“Defining a Table and Graph Style” in *SAS Output Delivery System: Procedures Guide*

**BORDERLEFTCOLOR=** *color*

specifies the color of the left border of the table.

**Restriction**
The `BORDERLEFTCOLOR=` attribute is valid only in markup family destinations, the RTF destination, printer family destinations, and the Measured RTF destination.

**Tip**
You might also need to specify the `BORDERLEFTWIDTH=` attribute to override the style in the ODS destination.

**See**
color style attribute value on page 383

**BORDERLEFTSTYLE=** *line-style*

specifies the line style of the left border of the specified table cell.

**line-style**
can be one of the following:

- DASHED
- DOTTED
- DOUBLE
- GROOVE
- HIDDEN
- INSET
- OUTSET
- RIDGE
- SOLID

**Restriction**
The `BORDERLEFTSTYLE=` attribute is valid only in markup family destinations, the RTF destination, and the Measured RTF destination.

**Tip**
You might also need to specify the `BORDERLEFTWIDTH=` attribute to override the style in the ODS destination.

**BORDERLEFTWIDTH=** *dimension*

specifies the width of the left border of a table or table cell.
Restriction  The BORDERLEFTWIDTH= attribute is valid only in markup family destinations, the RTF destination, printer family destinations, and the Measured RTF destination.

See  dimension attribute value on page 385

BORDERRIGHTCOLOR=\textit{color}  
specifies the color of the right border of a table or table cell.

Restriction  The BORDERRIGHTCOLOR= attribute is valid only in markup family destinations, the RTF destination, printer family destinations, and the Measured RTF destination.

Tip  You might also need to specify the BORDERRIGHTWIDTH= attribute to override the style in the ODS destination.

See  color style attribute value on page 383

BORDERRIGHTSTYLE=\textit{line-style}  
specifies the line style of the right border of the selected cell.

\textit{line-style}  
can be one of the following:

\begin{itemize}
  \item DASHED
  \item DOTTED
  \item DOUBLE
  \item GROOVE
  \item HIDDEN
  \item INSET
  \item OUTSET
  \item RIDGE
  \item SOLID
\end{itemize}

Restriction  The BORDERRIGHTSTYLE= attribute is valid only in markup family destinations, the RTF destination, and the Measured RTF destination.

Tip  You might also need to specify the BORDERRIGHTWIDTH= attribute to override the style in the ODS destination.

BORDERRIGHTWIDTH=\textit{dimension}  
specifies the width of the right border of the table.

Restriction  The BORDERRIGHTWIDTH= attribute is valid only in markup family destinations, printer family destinations, RTF destination, and the Measured RTF destination.

See  dimension attribute value on page 385

BORDERSPACING=\textit{dimension}  
specifies the vertical and horizontal thickness of the spacing between cells in a table.

Alias  CELLSPACING=
Default 0

Restriction The BORDERSPACING= attribute is valid in markup family destinations other than HTML5, printer family destinations, and the RTF destination.

Interaction If BORDERWIDTH= is nonzero, and if the background color of the table cells contrasts with the background color of the table, then the color of the table cell spacing is determined by the table's background.

See dimension attribute value on page 385

Examples “Creating a Stand-Alone Style” in SAS Output Delivery System: Procedures Guide

“Modifying the Default Style with the CLASS Statement” in SAS Output Delivery System: Procedures Guide

BORDERTOPCOLOR=color
specifies the color of the top border of a table or table cell.

Restrictions The BORDERTOPCOLOR= attribute is valid only in markup family destinations, printer family destinations, RTF destination, and the Measured RTF destination.

Tip You might also need to specify the BORDERTOPWIDTH= attribute to override the style in the ODS destination.

See color style attribute value on page 383

BORDERTOPSTYLE=line-style
specifies the line style of the top border of the specified table cell.

line-style
can be one of the following:

• DASHED
• DOTTED
• DOUBLE
• GROOVE
• HIDDEN
• INSET
• OUTSET
• RIDGE
• SOLID
Restrictions  The BORDERTOPSTYLE= attribute is valid only in markup family destinations, the RTF destination, and the Measured RTF destination.

For the RTF destination, specify the BORDERTOPSTYLE= attribute in conjunction with the BORDERTOPWIDTH= attribute to ensure that the style of the top border is the style that you specified.

Tip  You might also need to specify the BORDERTOPWIDTH= attribute to override the style in the ODS destination.

**BORDERTOPWIDTH=** *dimension*

specifies the width of the top border of the table or table cell.

Restriction  The BORDERTOPWIDTH= attribute is valid only in markup family destinations, printer family destinations, RTF destination, and the Measured RTF destination.

See  dimension attribute value on page 385

**BORDERWIDTH=** *dimension*

specifies the width of the table borders. The value of BORDERWIDTH= is applied to all four borders.

Restriction  The BORDERWIDTH= attribute is valid only in markup family destinations, printer family destinations, and the RTF destination.

Tip  Typically, when BORDERWIDTH=0, the ODS destination sets RULES=NONE (see the discussion about “RULES=rule-type” on page 378) and FRAME=VOID (see the discussion about “FRAME=frame-type” on page 367).

See  dimension attribute value on page 385

Examples  “Creating a Stand-Alone Style” in *SAS Output Delivery System: Procedures Guide*

          “Modifying the Default Style with the CLASS Statement” in *SAS Output Delivery System: Procedures Guide*

**CAPSTYLE=** *line-shape*

specifies the shape of the line at the end of a box whisker. *line-shape* can be one of the following:

- "BRACKET"
- "LINE"
- "NONE"
- "SERIF"

Requirement  You must enclose *line-shape* in quotation marks.

**CELLPADDING=**  

specifies the amount of white space on each of the four sides of the content in a table cell.

*dimension*

is a nonnegative number or the amount of white space on each of the four sides of the text in a table cell specified as a percentage of the table.
### Restrictions

The `CELLPADDING="attribute` is valid in markup family destinations other than HTML5, printer family destinations, and the RTF destination.

`CELLPADDING="attribute` is not valid in the HTML5 destination. All padding is done on the table cells.

### See

dimension attribute value on page 385

### Example

“Modifying the Default Style with the CLASS Statement” in SAS Output Delivery System: Procedures Guide

---

<table>
<thead>
<tr>
<th><strong>CLASS=&quot;string&quot;</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>specifies the name of the style sheet class to use in an HTML document for the table or table cell.</td>
</tr>
</tbody>
</table>

### Alias

`HTMLCLASS=`

### Restriction

The `CLASS="attribute` is valid only in markup family destinations.

### See

string attribute value on page 388

---

<table>
<thead>
<tr>
<th><strong>COLOR=&quot;color&quot;</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>specifies the color of the foreground in tables, table cells, or graphs, which is primarily the color of text.</td>
</tr>
</tbody>
</table>

### Alias

`FOREGROUND=`

### Restriction

The `COLOR="attribute` is valid only in markup family destinations, printer family destinations, and the RTF destination.

### Interaction

The `COLOR="attribute` is overridden by the `CBACK="option` in the SAS/GRAPH GOPTIONS statement.

### Tip

In a table, the `COLOR="attribute` affects only the text that is specified with the `PRETEXT="`, `POSTTEXT="`, `PREHTML="`, and `POSTHTML="` attributes. To alter the font for the text that appears in the table, set the attribute for a table cell.

### See

color style attribute value on page 383

### Examples

“Modifying the Default Style with the CLASS Statement” in SAS Output Delivery System: Procedures Guide

“Creating a Stand-Alone Style” in SAS Output Delivery System: Procedures Guide

---

<table>
<thead>
<tr>
<th><strong>CONNECT=&quot;connect-line-type&quot;</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>specifies the characteristics of a box plot connect line. <code>connect-line-type</code> can be one of the following:</td>
</tr>
<tr>
<td>&quot;MAX&quot;</td>
</tr>
<tr>
<td>&quot;MEAN&quot;</td>
</tr>
<tr>
<td>&quot;MEDIAN&quot;</td>
</tr>
<tr>
<td>&quot;MIN&quot;</td>
</tr>
<tr>
<td>&quot;Q1&quot;</td>
</tr>
</tbody>
</table>
• "Q3"

Requirement You must enclose connect-line-type in quotation marks.

CONTENTPOSITION=position
specifies the position, within the frame file, of the frames that display the contents and the page files. position can be one of the following:

LEFT
places the frames on the left.
Alias L

RIGHT
places the frames on the right.
Alias R

TOP
places the frames at the top.
Alias T

BOTTOM
places the frames at the bottom.
Alias B

Restriction The CONTENTPOSITION= attribute is valid only in markup family destinations.

See For information about the HTML files that ODS creates, see “HTML Links and References Produced by the HTML Destination” in SAS Output Delivery System: User's Guide.

CONTENTSCROLLBAR=YES | NO |AUTO
specifies whether to put a scroll bar in the frames in the frame file that display the contents and the page files. (For information about the HTML files that ODS creates, see “HTML Links and References Produced by the HTML Destination” in SAS Output Delivery System: User's Guide.)

YES
places a scroll bar in the frames in the frame file that display the contents and the page files.

NO
specifies not to put a scroll bar in the frames in the frame file that display the contents and the page files.

AUTO
specifies that the browser put a scroll bar on the table of contents frame only if the content in that panel is big enough to require scrolling.

Restriction The CONTENTSCROLLBAR= attribute is valid only in markup family destinations.

Tip Typically, CONTENTSCROLLBAR= is set to AUTO.
See For information about the HTML files that ODS creates, see “HTML Links and References Produced by the HTML Destination” in SAS Output Delivery System: User's Guide.

**CONTENTSIZEx=dimension | dimension % | ***
specifies the width of the frames in the frame file that display the contents and the page files.

*dimension*
is a nonnegative number or the width of the frames specified as a percentage of the entire display.

* specifies to use whatever space is left after displaying the body file as specified by the BODYSIZE= attribute.

**Restriction**
The CONTENTSIZE= attribute is valid only in markup family destinations.

**Requirement**
*dimension %* must be a positive number between 0 and 100.

**Tip**
If *dimension* is a nonnegative number, then the unit of measure is pixels.

See dimension attribute value on page 385

“BODYSIZE=dimension | dimension % | * ” on page 353

For information about the HTML files that ODS creates, see “HTML Links and References Produced by the HTML Destination” in SAS Output Delivery System: User's Guide.

**CONTENTTYPE="string"**
specifies the value of the content type for pages in an HTML document that is sent directly to a web server rather than to a file.

*string*
is the content type for the pages.

**Restriction**
The CONTENTTYPE= attribute is valid only in markup family destinations.

**Requirement**
*string* must be enclosed in quotation marks.

**Tip**
The value of *string* is usually "text/html".

See string attribute value on page 388

**Alias**
HTMLCONTENTTYPE=

**CONTRASTCOLOR=color**
specifies the alternate colors for maps. The alternate colors are applied to the blocks on region areas in block maps.

**Restriction**
The CONTRASTCOLOR= attribute is valid only in markup family destinations, printer family destinations, and the RTF destination.

See color style attribute value on page 383
DATASKIN=CRISP | GLOSS | MATTE | NONE | PRESSED | SHEEN
specifies the type of skin to apply to plots and charts (other than KPIs) to give them a raised, 3-D appearance.

The DATASKIN= style attribute is valid for the following plots and charts in the Graph Template Language:

• bar charts
• pie charts
• scatter plots
• waterfall charts

The DATASKIN= style attribute is valid for the following plots and charts in the SG procedures:

• bar charts
• scatter plots
• waterfall charts

Table 16.3  DATASKIN Values

<table>
<thead>
<tr>
<th>NONE</th>
<th>CRISP</th>
<th>GLOSS</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="NONE" /></td>
<td><img src="image" alt="CRISP" /></td>
<td><img src="image" alt="GLOSS" /></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>MATTE</th>
<th>PRESSED</th>
<th>SHEEN</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="MATTE" /></td>
<td><img src="image" alt="PRESSED" /></td>
<td><img src="image" alt="SHEEN" /></td>
</tr>
</tbody>
</table>

Restriction  In the first maintenance release of SAS 9.4 and later releases, the maximum number of skinned graphical elements is limited to 200 per plot in an overlay or prototype layout. When this limit is exceeded for a plot, the specified data skin is not applied to that plot. In that case, use the DATASKINMAX= option in your ODS GRAPHICS statement to increase the maximum limit.

DISPLAYOPTS="display-feature"
specifies one or more display features for ODS graphs. To specify multiple features, enclose the list of features in quotation marks (for example: displayopts="fill caps mean"). "display-feature" can be one of the following:

CAPS

displays caps at the ends of the whiskers.

Restriction  CAPS can be used only for box plots.

CONNECT

displays the line connecting multiple boxes.

Restriction  CONNECT can be used only for box plots.
FILL

displays filled boxes, bars, ellipses, and bands.

Restriction  
FILL can be used only for box plots, histograms, ellipses, and confidence bands.

MEAN

displays the mean symbol within a box.

Restriction  
MEAN can be used only for box plots.

MEDIAN

displays the median line within the box.

NOTCHES

displays notched boxes.

Restriction  
NOTCHES can be used only for box plots.

OUTLIERS

displays markers for the outliers.

Restriction  
OUTLIERS can be used only for box plots.

OUTLINE

displays outlined ellipses and bars.

Restriction  
OUTLINE can be used only for ellipses, bands, and histograms.

Requirement  
You must enclose "display-feature" in quotation marks.

DOCTYPE="string"

specifies the entire doctype declaration for the HTML document, including the opening "<!DOCTYPE" and the closing ">

string

is the doctype declaration.

Requirement  
string must be enclosed in quotation marks.

See  
string attribute value on page 388

Alias  
HTMLDOCTYPE=

Restriction  
The DOCTYPE= attribute is valid only in markup family destinations.

DROPSHADOW=ON | OFF

specifies whether the drop shadow color for text is displayed.

ENDCOLOR=color

specifies the final color used with a two- or three-color ramp.

See  
color style attribute value on page 383

FILLPATTERN=fillpattern-value

specifies the fill pattern to be displayed on the chart. The valid values are: S, E, L1, L2, L3, L4, L5, R1, R2, R3, R4, R5, X1, X2, X3, X4, and X5.
Restriction The FILLPATTERN= attribute is valid for bar charts only.

Tip To display these fill patterns on the bar chart through the style, you must also specify "fillpattern" as one of the DISPLAYOPTS in the GRAPHBAR style element.

See For a table of style elements and the style attributes that are valid in each one, see “Style Elements Affecting Template-Based Graphics” on page 321 and “Style Elements Affecting Device-Based Graphics” on page 329.

FILLRULEWIDTH=dimension
places a rule of the specified width into the space around the text (or entire cell if there is no text) in a table where white space would otherwise appear.

Restriction The FILLRULEWIDTH= attribute is valid only in printer family destinations.

Tip If no text is specified, then FILLRULEWIDTH= fills the space around the text with hyphen marks. For example: --this-- or this ------.

See dimension attribute value on page 385

FLYOVER="string"
specifies the text to show in a data tip for the table cell.

string is the text of the data tip.

Requirement string must be enclosed in quotation marks.

See string attribute value on page 388

Restriction The FLYOVER= attribute is valid only in markup family destinations and the PDF destination.

FONT=font-definition
specifies a font definition to use in tables, table cells, and graphs.

Restriction The FONT= attribute is valid only in markup family destinations, printer family destinations, and the RTF destination.

Tips For a table, the FONT= attribute affects only the text that is specified with the PRETEXT=, POSTTEXT=, PREHTML=, and POSTHTML= attributes. To alter the font for the text that appears in the table, set the attribute for a table cell.
If the system does not recognize the font specified, then it refers to the system's default font. This attribute does not accept concatenated fonts. SAS Graph Styles can only specify one font.

<table>
<thead>
<tr>
<th>See</th>
<th>font-definition attribute value on page 386</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example</td>
<td>“Modifying the Default Style with the CLASS Statement” in SAS Output Delivery System: Procedures Guide</td>
</tr>
</tbody>
</table>

**FONTFAMILY=**\"string-1<... string-n>\"

specifies the font to use in table cells and graphs. If you supply multiple fonts, then the destination device uses the first one that is installed on the system.

**string**

is the name of the font.

**Requirement**

*string* must be enclosed in quotation marks.

**See**

string attribute value on page 388

**Alias**

FONT_FACE=

**Restriction**

The FONTFAMILY= attribute is valid only in markup family destinations, printer family destinations, and the RTF destination.

**Tips**

For a table, the FONTFAMILY= attribute affects only the text that is specified with the PRETEXT=, POSTTEXT=, PREHTML=, and POSTHTML= attributes. To alter the font for the text that appears in the table, set the attribute for a table cell.

You cannot be sure what fonts are available to someone who is viewing the output in a browser or printing it on a high-resolution printer. Most devices support the following fonts: Times, Courier, Arial, Helvetica.

**Example**

“Creating a Stand-Alone Style” in SAS Output Delivery System: Procedures Guide

**FONTSIZE=**\tdimension | size\n
specifies the size of the font for tables, table cells, and graphs.

**dimension**

is a nonnegative number.

**Alias**

FONT_SIZE=

**Restriction**

If you specify a dimension, then specify a unit of measure. Without a unit of measure, the number becomes a relative size.

**See**

dimension attribute value on page 385

**size**

The value of size is relative to all other font sizes in the HTML document.

**Range**

1 to 7

**Restriction**

The FONTSIZE= attribute is valid only in markup family destinations, printer family destinations, and the RTF destination.
Tip
For a table, the FONTSIZE= attribute affects only the text that is specified with the PRETEXT=, POSTTEXT=, PREHTML=, and POSTHTML= attributes. To alter the font for the text that appears in the table, set the attribute for a table cell.

Example
“Creating a Stand-Alone Style” in SAS Output Delivery System: Procedures Guide

**FONTSTYLE=ITALIC | ROMAN | SLANT**
specifies the style of the font for tables, table cells, and graphs. In many cases, italic and slant map to the same font.

Alias
FONT_STYLE=

Restriction
The FONTSTYLE= attribute is valid only in markup family destinations, printer family destinations, and the RTF destination.

Tip
For a table, the FONTSTYLE= attribute affects only the text that is specified with the PRETEXT=, POSTTEXT=, PREHTML=, and POSTHTML= attributes. To alter the font for the text that appears in the table, set the attribute for a table cell.

Examples
“Creating a Stand-Alone Style” in SAS Output Delivery System: Procedures Guide
“Modifying the Default Style with the CLASS Statement” in SAS Output Delivery System: Procedures Guide

**FONTWEIGHT=weight**
specifies the font weight of tables, table cells, and graphs. weight is any of the following:

- MEDIUM
- BOLD
- DEMI_BOLD
- EXTRA_BOLD
- LIGHT
- DEMI_LIGHT
- EXTRA_LIGHT.

Alias
FONT_WEIGHT=

Restrictions
You cannot be sure what font weights are available to someone who is viewing the output in a browser or printing it on a high-resolution printer. Most devices support only MEDIUM and BOLD, and possibly LIGHT.

The FONTWEIGHT= attribute is valid only in markup family destinations, printer family destinations, and the RTF destination.

Tip
For a table, theFONTWEIGHT= attribute affects only the text that is specified with the PRETEXT=, POSTTEXT=, PREHTML=, and POSTHTML= attributes. To alter the font for the text that appears in the table, set the attribute for a table cell.
**Examples**

“Creating a Stand-Alone Style” in *SAS Output Delivery System: Procedures Guide*

“Creating a Stand-Alone Style” in *SAS Output Delivery System: Procedures Guide*

**FONTWIDTH=relative-width**

specifies the font width of tables, table cells, and graphs compared to the width of the usual design of the table, table cell, or graph. *relative-width* is any of the following:

- NORMAL
- COMPRESSED
- EXTRA_COMPRESSED
- NARROW
- WIDE
- EXPANDED

**Alias**

FONT_WIDTH=

**Restrictions**

Few fonts honor these values.

The FONTWIDTH= attribute is valid only in markup family destinations, printer family destinations, and the RTF destination.

**Tip**

For a table, the FONTWIDTH= attribute affects only the text that is specified with the PRETEXT=, POSTTEXT=, PREHTML=, and POSTHTML= attributes. To alter the font for the text that appears in the table, set the attribute for a table cell.

**Example**

“Creating a Stand-Alone Style” in *SAS Output Delivery System: Procedures Guide*

**FRAME=frame-type**

specifies the type of frame to use on a table. This table shows the possible values for *frame-type* and their meanings:

**Table 16.4 Frame-type Values**

<table>
<thead>
<tr>
<th>Value for frame-type</th>
<th>Frame Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABOVE</td>
<td>A border at the top</td>
</tr>
<tr>
<td>BELOW</td>
<td>A border at the bottom</td>
</tr>
<tr>
<td>BOX</td>
<td>Borders at the top, bottom, and both sides</td>
</tr>
<tr>
<td>HSIDES</td>
<td>Borders at the top and bottom</td>
</tr>
<tr>
<td>LHS</td>
<td>A border at the left side</td>
</tr>
<tr>
<td>RHS</td>
<td>A border at the right side</td>
</tr>
<tr>
<td>VOID</td>
<td>No borders</td>
</tr>
</tbody>
</table>
### Value for frame-type

<table>
<thead>
<tr>
<th>Value for frame-type</th>
<th>Frame Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>VSIDES</td>
<td>Borders at the left and right sides</td>
</tr>
</tbody>
</table>

**Restriction**

The FRAME= attribute is valid only in markup family destinations, printer family destinations, and the RTF destination.

**Example**

“Modifying the Default Style with the CLASS Statement” in *SAS Output Delivery System: Procedures Guide*

---

**FRAMEBORDER=ON | OFF**

specifies whether to put a border around the frame for an HTML file that uses frames.

- **ON** places a border around the frame for an HTML file that uses frames.
- **OFF** specifies not to put a border around the frame for an HTML file that uses frames.

**Restriction**

The FRAMEBORDER= attribute is valid only in markup family destinations.

---

**FRAMEBORDERWIDTH=dimension**

specifies the width of the border around the frames for an HTML file that uses frames.

**Restriction**

The FRAMEBORDERWIDTH= attribute is valid only in markup family destinations.

**See**

dimension attribute value on page 385

---

**FRAMESPACING=dimension**

specifies the width of the space between frames for HTML that uses frames.

**Restriction**

The FRAMESPACING= attribute is valid only in markup family destinations.

**See**

dimension attribute value on page 385

---

**GRADIENT_DIRECTION="YAXIS" | "XAXIS "**

specifies the direction of the gradient.

- **"YAXIS"** specifies a vertical gradient.
- **"XAXIS"** specifies a horizontal gradient.

---

**HEIGHT=dimension**

specifies the height of a table cell, graph, or graphics in an HTML document.

*dimension* is a nonnegative number.

**See**

dimension attribute value on page 385
Aliases

CELLHEIGHT=
OUTPUTHEIGHT=

Restrictions

The HEIGHT= option does not apply to output generated as a result of GRSEG (graph segment) output.

The HEIGHT= attribute is valid only in markup family destinations, printer family destinations, and the RTF destination.

Interaction

The YPIXELS= option in the SAS/GRAPH GOPTIONS statement overrides the HEIGHT= attribute.

Tip

HTML automatically sets cell height appropriately. You will seldom need to specify this attribute in the HTML destination.

HREFTARGET="target"

specifies the window or frame in which to open the target of the link. target is one of these values:

_blank
opens the target in a new, blank window. The window has no name.

Restriction Use lowercase letters to specify values for HREFTARGET.

_parent
opens the target in the window from which the current window was opened.

Restriction Use lowercase letters to specify values for HREFTARGET.

_search
opens the target in the browser's search pane.

Restrictions Only available in Internet Explorer 5.0 or later.

"name"
opens the target in the specified window or the frame.

Restrictions The HREFTARGET= attribute is valid only in markup family destinations.

Default _self

Requirement target must be enclosed in quotation marks.
**HTMLID=**"string"

specifies an ID for the table or table cell. The ID is for use by a Java Script.

*string*

is the ID text.

**Requirement**  *string* must be enclosed in quotation marks.

**See**  *string attribute value on page 388*

**Restriction**  The HTMLID= attribute is valid only in markup family destinations.

**HTMLSTYLE=**"string"

specifies individual attributes and values for a table or table cell in an HTML document.

*string*

is the name of an attribute or value.

**Requirement**  *string* must be enclosed in quotation marks.

**See**  *string attribute value on page 388*

**Restriction**  The HTMLSTYLE= attribute is valid only in markup family destinations.

**IMAGE=**"string"

specifies the image to appear in a graph. This image is positioned or tiled.

*string*

is the name of the image.

**Requirement**  *string* must be enclosed in quotation marks.

**See**  *string attribute value on page 388*

**Restriction**  The IMAGE= attribute is valid only in markup family destinations, printer family destinations, and the RTF destination.

**Interaction**  The BACK= and IMAGESTYLE=TILE options in the SAS/GRAPH GOPTIONS statement override the IMAGE= attribute.

**KPISKIN=BASIC | MODERN | NONE | ONYX | SATIN**

specifies the type of skin to apply to KPI charts to give them a raised, 3-D appearance.

**LINESTYLE=**pattern-number

specifies the pattern of a line. Valid pattern numbers range from 1 to 46. Not all pattern numbers have names. You must specify the line pattern by its number. *pattern-number* can be one of the following:
**Figure 16.1 Table of Line Patterns**

<table>
<thead>
<tr>
<th>Line Pattern</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Solid</td>
<td>1</td>
</tr>
<tr>
<td>ShortDash</td>
<td>2</td>
</tr>
<tr>
<td>MediumDash</td>
<td>4</td>
</tr>
<tr>
<td>LongDash</td>
<td>5</td>
</tr>
<tr>
<td>MediumDashShortDash</td>
<td>8</td>
</tr>
<tr>
<td>DashDashDot</td>
<td>14</td>
</tr>
<tr>
<td>DashDotDot</td>
<td>15</td>
</tr>
<tr>
<td>Dash</td>
<td>20</td>
</tr>
<tr>
<td>LongDashShortDash</td>
<td>26</td>
</tr>
<tr>
<td>Dot</td>
<td>34</td>
</tr>
<tr>
<td>ThinDot</td>
<td>35</td>
</tr>
<tr>
<td>ShortDashDot</td>
<td>41</td>
</tr>
<tr>
<td>MediumDashDotDot</td>
<td>42</td>
</tr>
</tbody>
</table>

**LINETHICKNESS=**`dimension`

specifies the thickness of a line.

See dimension attribute value on page 385

**LINKCOLOR=**`color`

specifies the color for the links in an HTML document that have not yet been visited.

Restriction The LINKCOLOR= attribute is valid only in markup family destinations, printer family destinations, and the RTF destination.

See color style attribute value on page 383

**LISTENTRYANCHOR=**`ON | OFF`

in an HTML document, the LISTENTRYANCHOR= attribute specifies whether to make the entry in the table of contents a link to the body file.

**ON**
specifies to make this entry in the table of contents a link to the body file.

**OFF**
specifies not to make this entry in the table of contents a link to the body file.

Restriction The LISTENTRYANCHOR= attribute is valid only in markup family destinations.

**LISTSTYLETYPE=**`bullet-type`

specifies the type of bullet to use for lists and the contents file. ODS uses bullets in the contents file.

`bullet-type`
is one of the following:
### MARGINBOTTOM=\textit{dimension}

specifies the bottom margin for the HTML document.

**Alias**

BULLET

**See**

string attribute value on page 388

**Restriction**

The MARGINBOTTOM= attribute is valid only in markup family destinations, printer family destinations, and the RTF destination.

**Tip**

If the orientation of a PDF document is changed after the PDF destination is opened and before the PDF destination is closed, any setting for margins is taken from the OPTIONS statement in place before the ODS PDF FILE= statement. If no OPTIONS statement is used to explicitly set the margins, the margin settings are retrieved from the SAS registry.

**See**

dimension attribute value on page 385

### MARGINLEFT=\textit{dimension}

specifies the left margin for the HTML document.
Alias  LEFTMARGIN=

Restriction The MARGINLEFT= attribute is valid only in markup family destinations, printer family destinations, and the RTF destination.

Tip If the orientation of a PDF document is changed after the PDF destination is opened and before the PDF destination is closed, any setting for margins is taken from the OPTIONS statement in place before the ODS PDF FILE= statement. If no OPTIONS statement is used to explicitly set the margins, the margin settings are retrieved from the SAS registry.

See dimension attribute value on page 385

MARGINRIGHT=\textit{dimension}

specifies the right margin for the HTML document.

Alias  RIGHTMARGIN=

Restriction The MARGINRIGHT= attribute is valid only in markup family destinations, printer family destinations, and the RTF destination.

Tip If the orientation of a PDF document is changed after the PDF destination is opened and before the PDF destination is closed, any setting for margins is taken from the OPTIONS statement in place before the ODS PDF FILE= statement. If no OPTIONS statement is used to explicitly set the margins, the margin settings are retrieved from the SAS registry.

See dimension attribute value on page 385

MARGINTOP=\textit{dimension}

specifies the top margin for the HTML document.

Alias  TOPMARGIN=

Restriction The MARGINTOP= attribute is valid only in markup family destinations, printer family destinations, and the RTF destination.

Tip If the orientation of a PDF document is changed after the PDF destination is opened and before the PDF destination is closed, any setting for margins is taken from the OPTIONS statement in place before the ODS PDF FILE= statement. If no OPTIONS statement is used to explicitly set the margins, the margin settings are retrieved from the SAS registry.

See dimension attribute value on page 385

MARKERSIZE=\textit{dimension}

specifies the marker size (both width and height).

See dimension attribute value on page 385

MARKERSYMBOL=\textit{marker-symbol}

specifies a marker symbol. \textit{marker-symbol} can be one of the following:
**Table of Marker Symbols**

- **NEUTRALCOLOR=**<br>specifies the middle color in a three-color ramp.<br>See [color style attribute value on page 383](#).

- **NOBREAKSPACE=ON | OFF**<br>specifies how to handle space characters in table cells.<br>**ON**<br>does not let SAS break a line at a space character.<br>**OFF**<br>lets SAS break a line at a space character if appropriate.<br>**Restriction**<br>The NOBREAKSPACE= attribute is valid in markup family destinations, printer family destinations, and the RTF destination.

- **OUTPUTHEIGHT=**<br>specifies the height of a graph.<br>See [dimension attribute value on page 385](#).

- **OUTPUTWIDTH=**<br>specifies the width of a graph.<br>See [dimension attribute value on page 385](#).

- **OVERHANGFACTOR=**<br>specifies an upper limit for extending the width of the column in an HTML document.<br>**Restriction**<br>The OVERHANGFACTOR= attribute is valid only in markup family and printer family destinations.<br>**Tips**<br>Typically, an overhang factor between 1 and 2 works well.<br>The HTML that is generated by ODS tries to ensure that the text in a column wraps when it reaches the requested column width. When the overhang factor greater than 1, the text can extend beyond the specified width.
**PADDING**=%dimension | dimension%

specifies the amount of white space between the content of the table cell and the border. The value of PADDING= applies to all four sides.

To change the padding of each side, use one or more of the following attributes:

- **PADDINGBOTTOM**= on page 375
- **PADDINGLEFT**= on page 375
- **PADDINGRIGHT**= on page 375
- **PADDINGTOP**= on page 375

**Restriction**
The PADDING= attribute is valid only in markup family destinations, printer family destinations, and the RTF destination.

**See**
dimension attribute value on page 385

**PADDINGBOTTOM**=%dimension | dimension%

specifies the amount of white space on the bottom of the content of the table cell.

**Default** 0

**Restriction**
The PADDINGBOTTOM= attribute is valid only in markup family destinations, printer family destinations, and the RTF destination.

**See**
dimension attribute value on page 385

**PADDINGLEFT**=%dimension | dimension%

specifies the amount of white space on the left side of the content of the table cell.

**Default** 0

**Restriction**
The PADDINGLEFT= attribute is valid only in markup family destinations, printer family destinations, and the RTF destination.

**See**
dimension attribute value on page 385

**PADDINGRIGHT**=%dimension | dimension%

specifies the amount of white space on the right side of the content of the table cell.

**Default** 0

**Restriction**
The PADDINGRIGHT= attribute is valid only in markup family destinations, printer family destinations, and the RTF destination.

**See**
dimension attribute value on page 385

**PADDINGTOP**=%dimension | dimension%

specifies the amount of white space on the top of the content of the table cell.

**Default** 0

**Restriction**
The PADDINGTOP= attribute is valid only in markup family destinations, printer family destinations, and the RTF destination.

**See**
dimension attribute value on page 385

**PAGEBREAKHTML**="string"

specifies HTML to place at page breaks in an HTML document.
string

is the HTML code used to place at page breaks.

Requirement  

string must be enclosed in quotation marks.

See  

string attribute value on page 388

Restriction  
The PAGEBREAKHTML= attribute is valid only in markup family destinations.

POSTHTML="string"

specifies the HTML code to place after the table or table cell.

string

is the HTML code to place after a table or table cell.

Requirement  

string must be enclosed in quotation marks.

See  

string attribute value on page 388

Restriction  
The POSTHTML= attribute is valid only in markup family destinations.

Example  

“Modifying the Default Style with the CLASS Statement” in SAS Output Delivery System: Procedures Guide

POSTIMAGE="string" | fileref

specifies an image to place after the table or table cell.

string

names a GIF or JPEG file. Use a simple filename, a complete path, or a URL.

Requirement  

string must be enclosed in quotation marks.

See  

string attribute value on page 388

fileref

is a reference that has been assigned to an external file. Use the FILENAME statement to assign a fileref.

See  

"Statements" in SAS Statements: Reference for information about the FILENAME statement.

Restriction  
The POSTIMAGE= attribute is valid only in markup family destinations, printer family destinations, and the RTF destination.

POSTTEXT="string"

specifies text to place after the table cell or table.

Restriction  
The POSTTEXT= attribute is valid only for markup family destinations, printer family destinations, and the RTF destination.

Requirement  

string must be enclosed in quotation marks.

See  

string attribute value on page 388

PREHTML="string"

specifies the HTML code to place before the table or table cell.
Restriction The PREHTML= attribute is valid only for markup family destinations.

See string attribute value on page 388

PREIMAGE="string" | fileref
specifies an image to place before the table or table cell.

string names a GIF or JPEG file. Use a simple filename, a complete path, or a URL.

Restriction When using the PREIMAGE= style attribute with the PRINTER destination, you must specify STARTPAGE=NO on the PRINTER family statement to display page numbers, times, dates, and titles. Without the STARTPAGE=NO option, preimages are treated like graphs and have no page numbers, times, dates, or titles displayed.

Requirement Enclose string in quotation marks.

See string attribute value on page 388

fileref is a reference that has been assigned to an external file. Use the FILENAME statement to assign a fileref. (For information about the FILENAME statement, see "Statements" in SAS Statements: Reference.)

Restriction The PREIMAGE= attribute is valid only in markup family destinations, printer family destinations, and the RTF destination.

PRETEXT="string"
specifies text to place before the table cell or table.

string text that is placed before the table cell or table.

Requirement Enclose string in quotation marks.

See string attribute value on page 388

Example “Customizing the Table of Contents” in SAS Output Delivery System: User’s Guide

Restriction The PRETEXT= attribute is valid only in markup family destinations, printer family destinations, and the RTF destination.

PROTECTSPECIALCHARS=ON | OFF | AUTO
specifies how less-than signs (<), greater-than signs (>), and ampersands (&) are interpreted in table cells. In HTML and other markup languages, these characters indicate the beginning of a markup tag, the end of a markup tag, and the beginning of the name of a file or character entity.

ON interprets special characters as the characters themselves. That is, when ON is in effect the characters are protected before they are passed to the HTML or other markup language destination so that the characters are not interpreted as part of the markup language. Using ON enables you to show markup language tags in the HTML document.
OFF
interprets special characters as markup language tags. That is, when OFF is in
effect, the characters are passed to the HTML or other markup language
destination without any protection so that the special characters are interpreted as
part of the markup language.

AUTO
interprets any string that starts with a < and ends with a > as a markup language
tag (ignoring spaces that immediately precede the <, spaces that immediately
follow the >, and spaces at the beginning and end of the string). In any other
string, AUTO protects the special characters from their markup language
meaning.

Restriction The PROTECTSPECIALCHARS= attribute is valid only in markup
family destinations, printer family destinations, and the RTF
destination.

RULES=rule-type
specifies the types of rules to use in tables. This table shows the possible values for
the RULES= attribute and their meanings:

<table>
<thead>
<tr>
<th>Value of RULES= Attribute</th>
<th>Locations of Rules</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALL</td>
<td>Between all rows and columns</td>
</tr>
<tr>
<td>COLS</td>
<td>Between all columns</td>
</tr>
<tr>
<td>GROUPS</td>
<td>Between the table header and the table and between the table and the table footer, if there is one</td>
</tr>
<tr>
<td>NONE</td>
<td>No rules anywhere</td>
</tr>
<tr>
<td>ROWS</td>
<td>Between all rows</td>
</tr>
</tbody>
</table>

Restriction The RULES= attribute is valid only in markup family destinations,
printer family destinations, and the RTF destination.

Example “Defining a Table and Graph Style” in SAS Output Delivery System:
Procedures Guide

STARTCOLOR=color
specifies the start fill color for a graph. It is used to create a gradient effect.

Note: You can have either a start and end gradient effect or no gradient effect. If you
specify a TRANSPARENCY level and you only specify the STARTCOLOR,
then the end color is completely transparent gradationally to the specified start
color.

Restriction The STARTCOLOR= attribute is valid only for the HTML destination.

See color style attribute value on page 383
**TAGATTR=**"**string**"

specifies text to insert into HTML.

*string*

is the text that is inserted into HTML tags.

**Requirements**  
*string* must be enclosed in quotation marks.

*string* must be valid HTML for the context in which the style element is created.

**Tip**

Many style elements are created between `<TD>` and `</TD>` tags. To determine how a style element is created, look at the source for the output.

**See**

*string* attribute value on page 388

**Restriction**

The TAGATTR= attribute is valid only in markup family destinations.

**TEXTALIGN=**alignment

specifies justification in tables, table cells, and graphs. In graphs, this option specifies the justification of the image specified with the IMAGE= statement. For example, this statement would produce a page number that is centered at the bottom of the page: **style PageNo from TitleAndFooters / textalign=c verticalalign=b**; This statement would produce a date in the body file that is left-justified at the top of the page: **style BodyDate from Date / textalign=l**; *alignment* can be one of the following:

**CENTER**

specifies center justification.

**Alias**  
C

**DEC**

specifies aligning the values by the decimal point.

**Alias**  
D

**Restriction**

Decimal alignment is supported for the printer family and RTF destinations only.

**LEFT**

specifies left justification.

**Alias**  
L

**RIGHT**

specifies right justification.

**Alias**  
R

**Restriction**

Not all contexts support RIGHT. If RIGHT is not supported, it is interpreted as CENTER.

**Alias**  
JUST=

**Restriction**

The TEXTALIGN= attribute is valid in markup family destinations other than HTML5, printer family destinations, and the RTF
destination. For the HTML5 destination, you might be able to use MARGINRIGHT=0 instead.

Tips

For the printer family destinations and the MARKUP destination, use the style attribute TEXTALIGN= with the style attribute VERTICALALIGN= in the style element PAGENO to control the placement of page numbers.

For printer family destinations and the MARKUP destination, control the placement of dates by using the style attribute TEXTALIGN= with the style attribute VERTICALALIGN= in the BODYDATE or DATE style element.

**TEXTDECORATION=presentation–options**

changes the visual presentation of the text. *presentation–options* can be one of the following:

**BLINK**

specifies that the text's visual presentation alternates rapidly between visible and invisible.

Restriction

TEXTDECORATION=BLINK is valid only in the HTML and RTF destinations.

**LINE_THROUGH**

specifies that a line is drawn through the text.

Restriction

TEXTDECORATION=LINE_THROUGH is valid only in the HTML destination, the printer family, the measured RTF destination, and the RTF destination.

**OVERLINE**

specifies that a line is drawn above the text.

Restriction

TEXTDECORATION=OVERLINE is valid only in the HTML destination and the printer family destinations.

**UNDERLINE**

specifies that a line is drawn below the text.

Restriction

TEXTDECORATION=UNDERLINE is valid only in the HTML destination, the printer family destinations, the measured RTF destination, and the RTF destination.

Tip

TEXTDECORATION= can be used with inline formatting and the ODS PDF statement to enhance PDF files.

Example

“Example 8: Formatting Cells Using the Textdecoration Style Attribute ” on page 219

**TEXTINDENT=n**

specifies the number of spaces that the first line of output is indented.

*n*

specifies the number of spaces to indent the output.

Alias

INDENT=}
The default value for XML is 2. For all other ODS destinations, the default value is 0.

The TEXTINDENT= attribute is valid only in the markup family destinations, the printer family destinations, and the RTF destination.

**TICKDISPLAY=**"INSIDE" | "OUTSIDE" | "ACROSS"
specifies the placement of all major and minor axis tick marks.

**TEXTJUSTIFY=**INTER_WORD | INTER_CHARACTER
specifies how to evenly distribute text.

**INTER_WORD**
specifies that the words are evenly distributed across the page.

**INTER_CHARACTER**
specifies that all characters are evenly distributed across a page.

**Tip** Use the TEXTJUSTIFY= style attribute with the TEXTALIGN=J (alias JUST=) style attribute.

**TRANSPARENCY=**dimension
specifies a transparency level for graphs. The values are 0.0 (opaque) to 1.0 (transparent).

The TRANSPARENCY= attribute is valid only in the HTML destination.

**URL=**"uniform-resource-locator"
specifies a URL to link to from the current cell.

The URL= attribute is valid only in markup family destinations, printer family destinations, and the RTF destination.

**Requirement** uniform-resource-locator must be enclosed in quotation marks.

**VERTICALALIGN=**BOTTOM | MIDDLE | TOP
specifies vertical justification for graphs and cells. In graphs, this option specifies the vertical justification of the image specified with IMAGE=. For example, this statement produces a page number that is centered at the bottom of the page: style PageNo from TitleAndFooters / textalign=c verticalalign=b;

This statement produces a date in the body file that is left-justified at the top of the page: style BodyDate from Date / textalign=l verticalalign=t;

**BOTTOM**
specifies bottom justification.

**Alias** B

**MIDDLE**
specifies center justification.

**Alias** M

**TOP**
specifies top justification.
<table>
<thead>
<tr>
<th>Alias</th>
<th>T</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alias</td>
<td>VJUST=</td>
</tr>
<tr>
<td>Restriction</td>
<td>The VERTICALALIGN= attribute is valid only in markup family destinations, printer family destinations, and the RTF destination.</td>
</tr>
<tr>
<td>Tips</td>
<td>For printer and markup family destinations, use the style attribute VERTICALALIGN= with the style attribute TEXTALIGN= in the style element PAGENO to control the placement of page numbers.</td>
</tr>
<tr>
<td></td>
<td>For printer and markup family destinations, control the placement of dates by using the style attribute VERTICALALIGN= with the style attribute TEXTALIGN= in the BODYDATE or DATE style element.</td>
</tr>
</tbody>
</table>

**VISITEDLINKCOLOR=color**

specifies the color for links that have been visited in an HTML document.

| Restriction | The VISITEDLINKCOLOR= attribute is valid only in markup family destinations. |
| See | color style attribute value on page 383 |

**WATERMARK=ON | OFF**

specifies whether to make the image that is specified by BACKGROUNDIMAGE= into a watermark. A watermark appears in a fixed position as the window is scrolled.

| ON | specifies to make the image that is specified by BACKGROUNDIMAGE= into a watermark. |
| OFF | specifies not to make the image that is specified by BACKGROUNDIMAGE= into a watermark. |

| Restriction | The WATERMARK= attribute is valid only in markup family destinations. |
| Tip | You can apply a watermark to output generated using the ODS TAGSETS.RTF destination by specifying a background image file. Note that the image is applied to the RTF document and not to a table or a table cell. |
| See | “BACKGROUNDIMAGE="string"” on page 351 |

**WHITESPACE=options**

specifies how the browser handles multiple whitespace characters and line breaks. options can be one of the following:

| NORMAL | specifies that white spaces are compressed and text wraps normally. |
| NOWRAP | specifies that white spaces are compressed and that text does not have line breaks. |
| PRE | specifies that white spaces are left intact and that text does not have line breaks. |
PRE_LINE specifies that white spaces are compressed, keeps line breaks that are in the text, and adds line breaks as needed.

PRE_WRAP specifies that white spaces are left intact and allows line breaking.

Default NORMAL

WIDTH=dimension specifies the width of a table cell, table, line, or a graph.

When used with graphs, the WIDTH= option must be specified as a pixel or percentage value. If a unit of measure is not specified with the dimension, then the value will be in pixels. If a unit of measure other than pixels or percentage is specified with the dimension, then the HEIGHT=dimension is not applied to the graph.

dimension is a nonnegative number.

See dimension attribute value on page 385

Aliases

CELLWIDTH=

OUTPUTWIDTH=

Restrictions

The WIDTH= option does not apply to output generated as a result of GRSEG (graph segment) output.

The WIDTH= attribute is valid only in markup family destinations, printer family destinations, and the RTF destination.

Interaction

The XPIXELS= option in the SAS/GRAPH GOPTIONS statement overrides the WIDTH= attribute.

Tips

A column of cells has the width of the widest cell in the column.

Use WIDTH=100% to make the table or graph as wide as the window that it is open in.

Style Attribute Values

color

is a string that identifies a color. A color is defined in the following ways:

• most of the color names that are supported by SAS/GRAPH. These names include the following:
  • a predefined SAS color (for example, blue or VIYG)
  • a red/green/blue (RGB) value (for example, CX0023FF)
  • a hue/light/saturation (HLS) value (for example, H14E162D)
  • a gray-scale value (for example, GRAYBB).
  • a red/green/blue transparency (RGBA) value (for example, a98FB9880)
• a cyan/magenta/yellow/black (CMYK) value (for example, FFFFFF00)

**Note:** RGBA color mode is not supported by Java devices. RGBA color mode is supported by ActiveX devices when the output is used in Microsoft applications.

• an RGB value with a leading number sign (#) rather than CX (for example, #0023FF).

• one of the colors that exist in the SAS session when the style is used:
  • DMSBLUE
  • DMSRED
  • DMSPINK
  • DMSGREEN
  • DMSCYAN
  • DMSYELLOW
  • DMSWHITE
  • DMSORANGE
  • DMSBLACK
  • DMSMAGENTA
  • DMSGRAY
  • DMSBROWN
  • SYSBACK
  • SYSSECB
  • SYSFORE

**Note:** Use these colors only when running SAS in the windowing environment.

• an English description of an HLS. Such descriptions use a combination of words to describe the lightness, the saturation, and the hue (in that order). Use the Color Naming System to form a color in the following ways:
  • combining a chromatic hue with a lightness, a saturation, or both
  • combining the achromatic hue gray with a lightness
  • combining the achromatic hue black or white without qualifiers

Use the words in the following table:

**Table 16.6 Hue/Light/Saturation (HLS) Values**

<table>
<thead>
<tr>
<th>Lightness</th>
<th>Saturation</th>
<th>Chromatic Hue</th>
<th>Achromatic Hue</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Blue</td>
<td>Black*</td>
</tr>
<tr>
<td>Very dark</td>
<td>Grayish</td>
<td>Purple</td>
<td></td>
</tr>
<tr>
<td>Dark</td>
<td>Moderate</td>
<td>Red</td>
<td></td>
</tr>
<tr>
<td>Medium</td>
<td>Strong</td>
<td>Orange</td>
<td>brown</td>
</tr>
</tbody>
</table>

*Note: Black* and **Gray** are used as placeholders for the actual achromatic hues.
### Lightness, Saturation, Chromatic Hue, Achromatic Hue

<table>
<thead>
<tr>
<th>Lightness</th>
<th>Saturation</th>
<th>Chromatic Hue</th>
<th>Achromatic Hue</th>
</tr>
</thead>
<tbody>
<tr>
<td>Light</td>
<td>Vivid</td>
<td>Yellow</td>
<td></td>
</tr>
<tr>
<td>Very light</td>
<td></td>
<td>Green</td>
<td>White</td>
</tr>
</tbody>
</table>

- Black and white cannot be combined with a lightness value or a saturation value.
- Gray cannot be combined with a saturation value.

Combine these words to form a wide variety of colors. Here are examples:

- light vivid green
- dark vivid orange
- light yellow

**Note:** The Output Delivery System first tries to match a color with a SAS/GRAPH color. Thus, although brown and orange are interchangeable in the table, if you use them as unmodified hues, then they are different. The reason for this is that ODS interprets them as SAS colors, which are mapped to different colors.

You can also specify hues that are intermediate between two neighboring colors. To do so, combine one of these adjectives with one of its neighboring colors:

- reddish
- orangish
- brownish
- yellowish
- greenish
- bluish
- purplish
- bluish purple
- reddish orange
- yellowish green

**Tips**
For a list of some valid colors, see [Link to Valid Colors to use with cascading style sheets](#).

To see how color names map to hexadecimal values, submit the following `REGISTRY` procedure code:

```sas
proc registry list startat="COLORNAMES";
run;
```

**See** [RBG Color Codes, HLS Color Codes, and Gray-Scale Color codes in SAS/GRAPH: Reference](#) for information about SAS/GRAPH colors.

**dimension**

is a whole number, a percentage, or a nonnegative number followed by one of these units of measure:
Table 16.7  Units of Measure for Dimension

<table>
<thead>
<tr>
<th>Unit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cm</td>
<td>Centimeters</td>
</tr>
<tr>
<td>em</td>
<td>Standard typesetting measurement unit for width</td>
</tr>
<tr>
<td>ex</td>
<td>Standard typesetting measurement unit for height</td>
</tr>
<tr>
<td>in</td>
<td>Inches</td>
</tr>
<tr>
<td>mm</td>
<td>Millimeters</td>
</tr>
<tr>
<td>pt</td>
<td>A printer’s point</td>
</tr>
</tbody>
</table>

Default  For the PRINTER destination, units of 1/150 of an inch

font-definition  is the name of a font, the font size, and font keywords. A font definition has this general format:

("font-face-1 <… ,font-face-n>“, font-size, keyword-list)

"font-face"  specifies the name of the font.

ODS styles can now use new TrueType fonts. All Universal Printers and many SAS/GRAPH devices use the FreeType library to render TrueType fonts for output in all of the operating environments that SAS software supports. In addition, by default, many SAS/GRAPH device drivers and all Universal Printers generate output using ODS styles, and these ODS styles use TrueType fonts. In addition to SAS Monospace and SAS Monospace Bold, 21 new TrueType fonts are made available when you install SAS:

• five Latin fonts compatible with Microsoft
• eight multilingual Unicode fonts
• eight monolingual Asian fonts

For more information about the TrueType fonts, see the section "Printing with SAS" in SAS Language Reference: Concepts.

Restriction  You must enclose multiple font-face in quotation marks. If you specify only one font and if its name does not include a space character, then omit the quotation marks.

Tip  If you specify more than one font, then the destination device uses the first one that is installed on the system.

font-size  specifies the size of the font. font-size is a dimension or a number without units of measure. If you specify a dimension, then specify a unit of measure. Without a unit of measure the number becomes a size that is relative to all other font sizes in the HTML document. For more information, see dimension attribute value on page 385.
keyword-list specifies the font weight, font style, and font width. Include one value for each, in any order. This table shows the keywords to use:

Table 16.8 Font Keywords

<table>
<thead>
<tr>
<th>Keywords for Font Weight</th>
<th>Keywords for Font Style</th>
<th>Keywords for Font Width</th>
</tr>
</thead>
<tbody>
<tr>
<td>MEDIUM</td>
<td>ITALIC</td>
<td>NORMAL</td>
</tr>
<tr>
<td>BOLD</td>
<td>ROMAN</td>
<td>COMPRESSED</td>
</tr>
<tr>
<td>DEMI_BOLD</td>
<td>SLANT</td>
<td>EXTRA_COMPRESSED</td>
</tr>
<tr>
<td>EXTRA_BOLD</td>
<td></td>
<td>NARROW</td>
</tr>
<tr>
<td>LIGHT</td>
<td></td>
<td>WIDE</td>
</tr>
<tr>
<td>DEMI_LIGHT</td>
<td></td>
<td>EXPANDED</td>
</tr>
<tr>
<td>EXTRA_LIGHT</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

* Few fonts honor these values.


format is a SAS format or a user-defined format.

integer | integer-list | integer-column-list
specifies a column variable that contains integer values, or a dynamic variable that refers to such a column variable.

integer specifies a single integer.

integer-list specifies a sequence of integer values, or a column variable that contains integer values, or a dynamic variable that refers to such a column variable or to a string.

integer-column-list specifies a sequence of column variables, or a column variable that contains column variables, or a dynamic variable that refers to such a column variable, or a dynamic variable that refers to a string containing a list of column variables. Values within the columns must be integers.

style-reference is a reference to an attribute that is defined in the current style or in the parent style (or beyond). The value used is the name of the style element followed by the name of an attribute, in parentheses, within that element. Style references have the following form:

style-attribute=target-style-element("target-style-attribute")

style-attribute specifies the name of the style attribute.
target-style-element
specifies the name of the style element that contains the style attribute that you want to reference.

target-style-attribute
specifies the style attribute with the value that you want to use.

Requirement
You must enclose target-style-attribute in quotation marks if it is a user-supplied style attribute.

See
“Understanding Style References ” in SAS Output Delivery System: Procedures Guide

Example

"string"
is a quoted character string.

user-defined-format
specifies a format created with the FORMAT procedure.

Restriction
user-defined-format can only be specified for data cells.
Part 5

Appendixes

Appendix 1

Example Programs ........................................... 391
Creating the DistrData Data Set

```sas
data distrdata;
  drop n;
  label Normal_x='Normal Random Variable'
      Exponential_x='Exponential Random Variable';
  do n=1 to 100;
    Normal_x=10*rannor(53124)+50;
    Exponential_x=ranexp(18746363);
    output;
  end;
run;
```

Creating the Univ ODS Document

```sas
ods document name=univ;

  title '100 Obs Sampled from a Normal Distribution';
  proc univariate data=distrdata noprint;
    var Normal_x;

    histogram Normal_x /normal(noprint) cbarline=grey name='normal';
  run;

  title '100 Obs Sampled from an Exponential Distribution';
  proc univariate data=distrdata noprint;
    var Exponential_x;

    histogram /exp(fill 1=3) cfill=yellow midpoints=0.05 to 5.55 by .25
           name='exp';
```
run;
ods document close;
title;
quit;
Here is the recommended reading list for this title. For a complete list of SAS publications, go to http://support.sas.com/publishing/index.html.

- SAS Graph Template Language: Reference
- SAS ODS Graphics: Procedures Guide
- SAS Graph Template Language: User's Guide
- Getting Started with the SAS Output Delivery System
- SAS Output Delivery System: Procedures Guide
- Base SAS Procedures Guide
- SAS Language Reference: Concepts
- SAS Data Set Options: Reference
- SAS Functions and CALL Routines: Reference
- SAS Statements: Reference
- SAS System Options: Reference
- Step-by-Step Programming with Base SAS
- SAS 9 ODS CSS Tip Sheet
- SAS 9 ODS EPUB Tip Sheet
- SAS 9 ODS Layout Tip Sheet
- SAS 9 ODS EXCELXP Tip Sheet
- SAS 9 Report Writing Interface Tip Sheet
- SAS 9 ODS List and Text Block Tip Sheet
- SAS offers instructor-led training and self-paced e-learning courses to help you get started with the SAS Output Delivery System and learn advanced techniques for the SAS Output Delivery System. For more information about the courses available, see http://support.sas.com/learn/.

The recommended reading list from SAS Press includes:

- Carpenter's Guide to Innovative SAS Techniques
Recommended Reading

- *Carpenter's Complete Guide to the SAS REPORT Procedure*
- *Learning SAS by Example: A Programmer's Guide*
- *The Little SAS Book: A Primer*
- *The Little SAS Book for Enterprise Guide*
- *Output Delivery System: The Basics and Beyond*
- *PROC TEMPLATE Made Easy: A Guide for SAS Users*
- *Statistical Graphics Procedures by Example: Effective Graphs Using SAS*
- *Statistical Graphics in SAS: An Introduction to the Graph Template Language and the Statistical Graphics Procedures*

For a complete list of SAS publications, go to [sas.com/store/books](http://sas.com/store/books). If you have questions about which titles you need, please contact a SAS Representative:

SAS Books
SAS Campus Drive
Cary, NC 27513-2414
Phone: 1-800-727-0025
Fax: 1-919-677-4444
Email: sasbook@sas.com
Web address: [sas.com/store/books](http://sas.com/store/books)
# Index

## Special Characters

<table>
<thead>
<tr>
<th>Character</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>NEW</em></td>
<td>129</td>
</tr>
<tr>
<td>ODS Object</td>
<td>131</td>
</tr>
<tr>
<td>(</td>
<td>128</td>
</tr>
<tr>
<td>)</td>
<td>130</td>
</tr>
</tbody>
</table>

## A

<table>
<thead>
<tr>
<th>term</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>absolute layout methods</td>
<td>161, 171, 176</td>
</tr>
<tr>
<td>ABSTRACT= style attribute</td>
<td>350</td>
</tr>
<tr>
<td>BACKGROUNDPOSITION= style attribute</td>
<td>351</td>
</tr>
<tr>
<td>ACTIVELINKCOLOR= style attribute</td>
<td>350</td>
</tr>
<tr>
<td>ASIS= style attribute</td>
<td>350</td>
</tr>
<tr>
<td>attribute selectors</td>
<td>52</td>
</tr>
<tr>
<td>cascading style sheets</td>
<td>52</td>
</tr>
<tr>
<td>AUDIO method</td>
<td>132</td>
</tr>
<tr>
<td>AUTOPLAY argument</td>
<td>134</td>
</tr>
<tr>
<td>AUDIO method</td>
<td>134</td>
</tr>
<tr>
<td>VIDEO method</td>
<td>194</td>
</tr>
</tbody>
</table>

## B

<table>
<thead>
<tr>
<th>term</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>BACKGROUND</td>
<td>61</td>
</tr>
<tr>
<td>BACKGROUND-COLOR</td>
<td>63</td>
</tr>
<tr>
<td>BACKGROUND= style attribute</td>
<td>351</td>
</tr>
<tr>
<td>BACKGROUNDCOLOR= style attribute</td>
<td>351</td>
</tr>
<tr>
<td>BACKGROUNDIMAGE= style attribute</td>
<td>351</td>
</tr>
<tr>
<td>BACKGROUNDREPEAT= Style Attribute</td>
<td>352</td>
</tr>
<tr>
<td>BODY_END method</td>
<td>136</td>
</tr>
<tr>
<td>BODY_START method</td>
<td>135</td>
</tr>
<tr>
<td>BODYS卷BAR= style attribute</td>
<td>352</td>
</tr>
<tr>
<td>BODYSIZE= style attribute</td>
<td>353</td>
</tr>
<tr>
<td>BORDER side-STYLE</td>
<td>65</td>
</tr>
<tr>
<td>BORDER side-WIDTH</td>
<td>66</td>
</tr>
<tr>
<td>BORDER STYLE</td>
<td>66</td>
</tr>
</tbody>
</table>

CSS style property 66
BORDER WIDTH
CSS style property 67
BORDER-COLOR
CSS style property 64
BORDER-side-COLOR
CSS style property 64
BORDER-SPACING
CSS style property 68
BORDERBOTTOMCOLOR= style attribute 353
BORDERBOTTOMSTYLE= style attribute 353
BORDERBOTTOMWIDTH= style attribute 354
BORDERCOLLAPSE= style attribute 354
BORDERCOLOR= style attribute 354
BORDERCOLORDARK= style attribute 354
BORDERCOLORLIGHT= style attribute 355
BORDERLEFTCOLOR= style attribute 355
BORDERLEFTSTYLE= style attribute 355
BORDERLEFTWIDTH= style attribute 355
BORDERRIGHTCOLOR= style attribute 356
BORDERRIGHTSTYLE= style attribute 356
BORDERRIGHTWIDTH= style attribute 356
BORDERSPACING= style attribute 356
BORDERTOPCOLOR= style attribute 357
BORDERTOPSTYLE= style attribute 357
BORDERTOPWIDTH= style attribute 358
BORDERWIDTH= style attribute 358
BOTTOMMARGIN= style attribute 372
BULLET= style attribute 371
Index

C
CAPSTYLE= style attribute 358

cascading style sheets
  attribute selectors 52
class selectors 50
combinators 53
element selectors 50
ID selectors 52
ODS 41, 42, 43
pseudo-class selectors 51
universal selector 51
CELL_END method 141
CELL_START method 136
CELLHEIGHT= style attribute 368
CELLPADDING= style attribute 358
CELLSPACING= style attribute 356
CELLWIDTH= style attribute 383
class selectors
cascading style sheets 50
CLASS= style attribute 359
CLEAR option
  FOOTNOTE method 153
title method 189
COLOR
  CSS style property 68
COLOR= style attribute 359
COLUMN option
  REGION method, Gridded 181
COLUMN_GUTTER option
  LAYOUT_GRIDDED method 165
COLUMN_SPAN
  FORMAT_CELL method 145
COLUMN_SPAN option
  CELL_START method 137
COLUMN_SPAN: option
  REGION method, Gridded 181
COLUMN_WIDTHS option
  LAYOUT_GRIDDED method 166
COLUMNS option
  LAYOUT_GRIDDED method 165
combinator operator
cascading style sheets 42
combinators
cascading style sheets 53
CONNECT= style attribute 359
CONTENTPOSITION= style attribute 360
CONTENTSCROLLBAR= style attribute 360
CONTENTSIZE= style attribute 361
CONTEETYTYPE= style attribute 361
CONTRASTCOLOR= style attribute 361
CSS
  ODS document object model 45
  CSS declaration block
cascading style sheets 43
CSS rule set
cascading style sheets 42

D
DATASKIN= style attribute 362
DCL statement
  ODS Object 130
DECLARE statement
  ODS Object 130
DELETE method 142
DISPLAYOPTS= style attribute 362
DOCTYPE= style attribute 363
DOCUMENT procedure
  WHERE expressions with 21
Documents window
  pop-up menu 7
DOM
  ODS document object model 45
  DOM element
cascading style sheets 42
DROPSHADOW= style attribute 363

E
element selectors
cascading style sheets 50
ENDCOLOR= style attribute 363
EXPRESSION
  CSS dynamic style property 57

F
FILE argument
  AUDIO method 133
  IMAGE method 160
  VIDEO method 192
FILLPATTERN= style attribute 363
FILLRULEWIDTH= style attribute 364
FLYOVER= style attribute 364
FONT
  CSS style property 70
  FONT-FAMILY
    CSS style property 69
  FONT-SIZE
    CSS style property 69
  FONT-STYLE
    CSS style property 70
  FONT-WEIGHT
    CSS style property 70
  FONT_FACE= style attribute 365
  FONT= style attribute 364
  FONTFAMILY= style attribute 365
  FONTSIZE= style attribute 365
  FONTSTYLE= style attribute 366
  FONTWIGHT= style attribute 366
INDEX

INDEX

FOOT_WIDTH= style attribute 367
FOOT_END method 143
FOOT_START method 142
FOOTNOTE method 152
FOREGROUND_COLOR= style attribute 359
FORMAT
  CSS dynamic style property 57
  FORMAT_CELL method 144
  FORMAT_TEXT method 149
FRAME= style attribute 367
FRAME_BORDER= style attribute 368
FRAME_BORDER_WIDTH= style attribute 368
FRAME_SPACING= style attribute 368
GRADIENT_DIRECTION= style attribute 368
  gridded layout methods 171, 180
GRIDDED layout methods 165
HEAD_END method 156
HEAD_START method 155
HEIGHT argument
  VIDEO method 194
HEIGHT option
  CELL_START method 138
  FORMAT_CELL method 146
  FORMAT_TEXT method 149
HEAD_END method 156
HEAD_START method 155
HEIGHT argument
  VIDEO method 194
HEIGHT option
  CELL_START method 138
  FORMAT_CELL method 145
  FORMAT_TEXT method 149
  LAYOUT_ABSOLUTE method 161, 165
  LAYOUT_END method 171
LEFT_MARGIN= style attribute 372
LINE method 171
LINE_STYLE= style attribute 370
LINE_THICKNESS= style attribute 371
LINK_COLOR= style attribute 371
LISTENTRY_ANCHOR= style attribute 371
LIST_STYLE_TYPE= style attribute 371
LOOP argument
  AUDIO method 134
  VIDEO method 194
MARGIN_BOTTOM= style attribute 372
MARGIN_LEFT= style attribute 372
MARGIN_RIGHT= style attribute 373
MARGIN_TOP= style attribute 373
MARKER_SIZE= style attribute 373
MARKER_SYMBOL= style attribute 373
media 132, 191
  methods 142, 157, 160, 171
  AUDIO 132
  BODY_END 136
  BODY_START 135
  CELL_END 141
  CELL_START 136
  DELETE 142
  FOOT_END 143
  FOOT_START 142
  FOOTNOTE 152
  FORMAT_CELL 144
  FORMAT_TEXT 149
  HEAD_END 156
  HEAD_START 155
  HREF 157
  IMAGE 160
  LAYOUT_ABSOLUTE 171
  LAYOUT_GRIDDED 161, 165, 171
LINE 171
NOTES 173
PAGE 176
REGION 176, 180
ROW_END 184
ROW_START 185
TABLE_END 186
TABLE_START 187
TITLE 188
VIDEO 191
MUTED argument
  AUDIO method 135
  VIDEO method 194

N
NAME option
  TABLE_START method 188
NEUTRALCOLOR= style attribute 374
NOBREAKSPACE= style attribute 374
NOTES method 173

O
object–reference argument
  _NEW_ Operator statement 132
DECLARE statement 130
ODS
  attribute selectors 52
cascading style sheets 42, 43
class selectors 50
combinators 53
element selectors 50
ID selectors 52
pseudo-class selectors 51
universal selector 51
ODS document object model
  DOM 45
ODS documents
  listing 25
  opening 25
ODS object 130, 131
ODSOUT argument
  _NEW_ Operator statement 132
DECLARE statement 130
options
  BODY_START method 135
ORDER_TYPE option
  LAYOUT_GRIDDED method 167
OUTPUTHEIGHT= style attribute 368, 374
OUTPUTWIDTH= style attribute 374, 383
OVERHANGFACTOR= style attribute 374

P
PADDING
  CSS style property 71
PADDING-side
  CSS style property 72
PADDING= style attribute 375
PADDINGBOTTOM= style attribute 375
PADDINGLEFT= style attribute 375
PADDINGRIGHT= style attribute 375
PADDINHTOP= style attribute 375
PAGE method 176
page methods 152, 176, 188
PAGEBREAKHTML= style attribute 375
POSTER argument
  VIDEO method 194
POSTHTML= style attribute 376
POSTIMAGE= style attribute 376
POSTTEXT= style attribute 376
PREHTML= style attribute 376
PREIMAGE= style attribute 377
PRELOAD argument
  AUDIO method 134
  VIDEO method 194
PRETEXT= style attribute 377
procedures
  style templates with 265
PROTECTSPECIALCHARS= style attribute 377
pseudo-class selectors
cascading style sheets 51

R
REGION method 176, 180
report writing interface
  RWI 130, 131
RESOLVE
  CSS dynamic style property 57
RIGHTMARGIN= style attribute 373
ROW option
  REGION method, Gridded 182
ROW option
  ROW_START method 185
ROW_END method 184
ROW_GUTTER option
  LAYOUT_GRIDDED method 168
ROW_HEIGHTS option
  LAYOUT_GRIDDED method 168
ROW_SPAN option
  CELL_START method 140
  FORMAT_CELL method 147
REGION method, Gridded 182
ROW_START method 185
ROWS option
  LAYOUT_GRIDDED method 167
RULES= style attribute 378
S
SIZE option
   LINE method 172
SPLIT option
   FORMAT_TEXT method 151
   NOTE method 175
START option
   FOOTNOTE method 154
   TITLE method 190
STARTCOLOR= style attribute 378
style attributes
   color 383
data values 339
definition 259
dimension 385
font definition 386
format 387
reference 387
table of 339
values of 339
style elements
   definition 259
   inheritances of 311
TABULATE procedure 259
style property
cascading style sheets 43
style selector
cascading style sheets 42
style templates
   definition of 259
   procedures with 265
   viewing contents of 269
SYMGET
   CSS dynamic style property 57

T
table methods 135, 136, 141, 142, 143, 144, 155, 156, 184, 185, 186, 187
   TABLE_END method 186
   TABLE_START method 187
   TABULATE procedure
      style elements 259
   TAGATTR= style attribute 379
text methods 149, 173
   TEXTALIGN= style attribute 379
   TEXTDECORATION= style attribute 380
   TEXTINDENT= style attribute 380
   TEXTJUSTIFY= style attribute 381
   TICKDISPLAY= style attribute 381
   TITLE method 188
   TOPMARGIN= style attribute 373
   TRANSPARENCY= style attribute 381
   TYPE argument
      AUDIO method 133
      VIDEO method 193
   TYPE option
      ROW_START method 186

U
universal selector
cascading style sheets 51
URL= style attribute 381

V
   VERTICALALIGN= style attribute 381
   VIDEO method 191
   VISITEDLINKCOLOR= style attribute 382
   VJUST= style attribute 381

W
   WATERMARK= style attribute 382
   WHERE expressions
      DOCUMENT procedure with 21
   WHITESPACE= style attribute 382
   WIDTH argument
      VIDEO method 193
   WIDTH option
      CELL_START method 141
      FORMAT_CELL method 149
      LAYOUT_ABSOLUTE method 163
      LAYOUT_GRIDDIED method 169
      REGION method 178
      REGION method, Gridded 183
   WIDTH= style attribute 383

X
   X option
      LAYOUT_ABSOLUTE method 163
      LAYOUT_GRIDDIED method 170
      REGION method 178

Y
   Y option
      LAYOUT_ABSOLUTE method 164
      LAYOUT_GRIDDIED method 170
      REGION method 179
Gain Greater Insight into Your SAS® Software with SAS Books.

Discover all that you need on your journey to knowledge and empowerment.

support.sas.com/bookstore for additional books and resources.