

Paper 297-2011

Unveiling the Power of Cascading Style Sheets (CSS) in ODS

Kevin D. Smith, SAS Institute Inc., Cary, NC

ABSTRACT

Prior to SAS® 9.2, PROC TEMPLATE was the only choice for writing ODS styles. SAS 9.2 added the capability of writing styles using Cascading Style Sheets (CSS) syntax. However, the accepted CSS syntax was primarily a remapping of PROC TEMPLATE elements and attributes to CSS classes and properties. While this remapped syntax was a step in the right direction, no new capabilities were added to the styles themselves. CSS support in SAS® 9.3 takes a bigger leap toward the power of CSS seen on the Internet today. While most people think of CSS as an HTML-only technology, it is a style syntax that can also be applied to other ODS outputs such as PDF, RTF, and ExcelXP. In fact, CSS in SAS 9.3 works with any of the ODS outputs that use styles. This paper uncovers the hidden power of CSS in SAS 9.3 and shows you things that were never before possible in previous versions of SAS as well as directions SAS will be taking in the future.

INTRODUCTION

Prior to SAS 9.2, the only way to write styles for ODS was to use PROC TEMPLATE¹. These styles consisted of a collection of style elements, each of which can be applied to a part of a report by using the element names used by ODS or by specifying a style element from one of the reporting procedures (PROC PRINT, PROC REPORT, PROC TABULATE) or a table template. The reporting procedures and table templates also have mechanisms for dynamically specifying style elements and attributes for traffic-lighting or just for aesthetic purposes. However, the style definitions themselves were primarily a collection of static elements and attributes².

In SAS 9.2, the playing field expanded with the inclusion of a Cascading Style Sheet (CSS) parser in ODS. This feature made it possible to write styles using CSS syntax rather than the proprietary PROC TEMPLATE STYLE syntax. While the syntax of CSS is more commonly known than PROC TEMPLATE STYLE syntax, it still remained a collection of static style elements and attributes. The implementation of CSS in ODS in SAS 9.2 ended at just that, a parser. It merely read CSS formatted styles and converted them, internally, into PROC TEMPLATE styles. However, that was an important step in the evolution of styles in SAS and led to further developments in SAS 9.3.

While on the surface, ODS styles in SAS 9.3 might look and behave the way they always have, under the covers they have been completely overhauled. The CSS implementation in ODS is no longer merely a CSS parser; it is a full-blown CSS selector engine that goes beyond merely linking to a CSS file from HTML output. CSS is implemented within ODS so that it works with all output types that use styles. That means there is no longer just one style element associated with a part of a report. It is possible to apply multiple style elements to one region of a report using CSS selectors. Because CSS selectors are "context based", you can apply styles to parts of a report based on the type of element, its position relative to other elements, and attribute values. Needless to say, the style selection capabilities are far more dynamic and powerful than ever before.

NOTE. Although the new style engine in ODS is production quality, many of the new CSS selector features are still considered preproduction.

Now that we have summarized the recent history of ODS styles and their evolution into an implementation of CSS, let's work through some examples that show how to use them.

THE PROC TEMPLATE STYLES METHOD

Before we jump into the world of CSS, let's take a minute to review the traditional way of writing style definitions in PROC TEMPLATE. Below is a simple style definition defined using PROC TEMPLATE. It does not define all of the style elements used by ODS, but it does create a simple, usable style.

```
proc template;
  define style mystyle;

    /* Body */
    class body /
```

¹ In HTML and SAS® Enterprise Guide® output, it was possible to link to a Cascading Style Sheet, but this technique uses styles outside of the SAS system.

² Exceptions are the expression, resolve, and symget functions as attribute values for macros and expressions.

Unveiling the Power of Cascading Style Sheets (CSS) in ODS, continued

```
        backgroundcolor = white
        color = black
    ;

/* Tables */
class table /
    frame = box
    rules = groups
    borderwidth = 1px
    borderstyle = solid
    bordercolor = black
    borderspacing = 0
    bordercollapse = collapse
    cellpadding = 5px
;
class data, header, rowheader /
    fontfamily = '"Lucida Grande", Arial, sans-serif'
    fontsize = 11pt
    backgroundcolor = #f0f0f0
    color = black
;
class header, rowheader /
    fontweight = bold
    textalign = left
    fontsize = 12pt
    backgroundcolor = #d0d0d0
;

/* System Title and Footers */
class systemtitle, systemfooter /
    fontfamily = '"Lucida Grande", Arial, sans-serif'
;

/* Page number and date (for printer) */
class pageno, bodydate /
    fontfamily = '"Lucida Grande", Arial, sans-serif'
;

end;
run;
```

We can generate some output using the above style definition with the code below.

```
ods html style=mystyle; ods pdf style=mystyle;

proc print data=sashelp.class;
run;

ods _all_ close;
```

Unveiling the Power of Cascading Style Sheets (CSS) in ODS, continued

Here is what the output looks like in HTML and PDF, respectively.

HTML

The SAS System

Obs	Name	Sex	Age	Height	Weight
1	Alfred	M	14	69.0	112.5
2	Alice	F	13	56.5	84.0
3	Barbara	F	13	65.3	98.0
4	Carol	F	14	62.8	102.5
5	Henry	M	14	63.5	102.5
6	James	M	12	57.3	83.0
7	Jane	F	12	59.8	84.5
8	Janet	F	15	62.5	112.5
9	Jeffrey	M	13	62.5	84.0
10	John	M	12	59.0	99.5
11	Joyce	F	11	51.3	50.5
12	Judy	F	14	64.3	90.0
13	Louise	F	12	56.3	77.0
14	Mary	F	15	66.5	112.0
15	Philip	M	16	72.0	150.0
16	Robert	M	12	64.8	128.0
17	Ronald	M	15	67.0	133.0
18	Thomas	M	11	57.5	85.0
19	William	M	15	66.5	112.0

PDF

The SAS System

10:36 Wednesday, November 17, 2010 1

Obs	Name	Sex	Age	Height	Weight
1	Alfred	M	14	69.0	112.5
2	Alice	F	13	56.5	84.0
3	Barbara	F	13	65.3	98.0
4	Carol	F	14	62.8	102.5
5	Henry	M	14	63.5	102.5
6	James	M	12	57.3	83.0
7	Jane	F	12	59.8	84.5
8	Janet	F	15	62.5	112.5
9	Jeffrey	M	13	62.5	84.0
10	John	M	12	59.0	99.5
11	Joyce	F	11	51.3	50.5
12	Judy	F	14	64.3	90.0
13	Louise	F	12	56.3	77.0
14	Mary	F	15	66.5	112.0
15	Philip	M	16	72.0	150.0
16	Robert	M	12	64.8	128.0
17	Ronald	M	15	67.0	133.0
18	Thomas	M	11	57.5	85.0
19	William	M	15	66.5	112.0

So far the output looks very reasonable. Looking back at the style definition, very few style elements were actually specified. They are BODY, TABLE, HEADER, ROWHEADER, DATA, SYSTEMTITLE, SYSTEMFOOTER, PAGENO, and BODYDATE. For most reports, these nine³ style elements achieve satisfactory results.

For those who are not yet familiar with the CLASS statement introduced in SAS 9.2, let's cover that briefly. The CLASS statement was directly influenced by CSS classes and provides alternate syntax for concepts that already existed in SAS before version 9.2.

The SAS 9.2 statement

```
CLASS foo / color = red;
```

Is equivalent to this traditional statement

```
STYLE foo FROM foo / color = red;
```

The one extra feature of the CLASS statement that you cannot do with a traditional STYLE statement is to specify multiple style element names to apply the attributes to. You can see in the example above that we applied font and color information to data, header, and rowheader simultaneously in the CLASS statement by supplying all three names separated by commas rather than just specifying a single style element name. This syntax is very reminiscent of CSS, which allows you to do the same thing.

Another thing you might have noticed is that HEADER and ROWHEADER were specified in more than one CLASS statement. When this happens, the attributes from both CLASS statements are merged together. If they have conflicting style attributes, the style attributes that come later take precedence. This is called cascading inheritance and is the origin of the term "cascading" in the name "cascading style sheets."

As you can see, even traditional style definitions created using PROC TEMPLATE already started to borrow concepts from CSS in SAS 9.2. Adding these CSS features made it natural to embed a CSS parser into ODS to allow the use of CSS files directly from the ODS statement, which is where we start in the next section.

³ The PAGENO and BODYDATE elements have no effect on non-paged output such as HTML.

Unveiling the Power of Cascading Style Sheets (CSS) in ODS, continued

THE CSS METHOD

SAS 9.2 added the ability to read CSS formatted files in addition to PROC TEMPLATE styles. The capabilities were somewhat limited in that version in that it could read only CSS files that were formatted similar to what ODS HTML could generate. Below is a partial listing of the CSS content generated by ODS HTML from the SAS program in the previous section. You can see that it closely parallels the PROC TEMPLATE style defined in that section.

```
.body {
  background-color: #FFFFFF;
  color: #000000;
}
.bodydate {
  font-family: "Lucida Grande", Arial, sans-serif;
}
.data {
  background-color: #F0F0F0;
  color: #000000;
  font-family: "Lucida Grande", Arial, sans-serif; font-size: 11pt;
}
.header {
  background-color: #D0D0D0;
  color: #000000;
  font-family: "Lucida Grande", Arial, sans-serif;
  font-size: 12pt;
  font-weight: bold;
  text-align: left;
}
.pageno {
  font-family: "Lucida Grande", Arial, sans-serif;
}
.rowheader {
  background-color: #D0D0D0;
  color: #000000;
  font-family: "Lucida Grande", Arial, sans-serif;
  font-size: 12pt;
  font-weight: bold;
  text-align: left;
}
.systemfooter {
  font-family: "Lucida Grande", Arial, sans-serif;
  font-size: 13pt;
  font-weight: bold;
}
.systemtitle {
  font-family: "Lucida Grande", Arial, sans-serif;
  font-size: 13pt;
  font-weight: bold;
}
.table {
  border: 1px solid #000000;
  border-collapse: collapse;
  border-spacing: 0px;
}
```

It is possible to use this CSS code as a style for ODS just as you can a PROC TEMPLATE style. You specify the CSS style with the CSSSTYLE= option. The SAS program below demonstrates the use of the CSSSTYLE= option.

```
ods html cssstyle='ex1.css';
ods pdf cssstyle='ex1.css';

proc print data=sashelp.class;
run;

ods _all_ close;
```

Unveiling the Power of Cascading Style Sheets (CSS) in ODS, continued

Note that whereas CSS is generally thought of as an HTML-only technology, the CSSSTYLE= option works on any ODS output that supports the STYLE= option. Here is the output from the above program.

HTML						PDF					
The SAS System						The SAS System					
Obs	Name	Sex	Age	Height	Weight	Obs	Name	Sex	Age	Height	Weight
1	Alfred	M	14	69.0	112.5	1	Alfred	M	14	69.0	112.5
2	Alice	F	13	56.5	84.0	2	Alice	F	13	56.5	84.0
3	Barbara	F	13	65.3	98.0	3	Barbara	F	13	65.3	98.0
4	Carol	F	14	62.8	102.5	4	Carol	F	14	62.8	102.5
5	Henry	M	14	63.5	102.5	5	Henry	M	14	63.5	102.5
6	James	M	12	57.3	83.0	6	James	M	12	57.3	83.0
7	Jane	F	12	59.8	84.5	7	Jane	F	12	59.8	84.5
8	Janet	F	15	62.5	112.5	8	Janet	F	15	62.5	112.5
9	Jeffrey	M	13	62.5	84.0	9	Jeffrey	M	13	62.5	84.0
10	John	M	12	59.0	99.5	10	John	M	12	59.0	99.5
11	Joyce	F	11	51.3	50.5	11	Joyce	F	11	51.3	50.5
12	Judy	F	14	64.3	90.0	12	Judy	F	14	64.3	90.0
13	Louise	F	12	56.3	77.0	13	Louise	F	12	56.3	77.0
14	Mary	F	15	66.5	112.0	14	Mary	F	15	66.5	112.0
15	Philip	M	16	72.0	150.0	15	Philip	M	16	72.0	150.0
16	Robert	M	12	64.8	128.0	16	Robert	M	12	64.8	128.0
17	Ronald	M	15	67.0	133.0	17	Ronald	M	15	67.0	133.0
18	Thomas	M	11	57.5	85.0	18	Thomas	M	11	57.5	85.0
19	William	M	15	66.5	112.0	19	William	M	15	66.5	112.0

The output above does not look quite right using the round-tripped CSS code. The padding on the table cells is missing. This is caused by a difference in the way that traditional PROC TEMPLATE styles and CSS do padding within tables. PROC TEMPLATE styles generally use CELLPADDING= on the Table style element for padding within table cells. This is because PROC TEMPLATE styles were primarily based on the HTML model of styling at the time it was written. CSS does not have a concept of CELLPADDING= for tables. CSS simply allows you to specify padding on any element. Since we used CELLPADDING= in our original PROC TEMPLATE Style, that attribute did not get translated into the corresponding CSS code. To remedy this, we need to change the CELLPADDING= in our Table style element to PADDING= on the DATA, HEADER, and ROWHEADER elements. Here is the new style code with the padding changes incorporated⁴.

```
proc template;
define style mystyle;
  /* Body */
  class body /
    backgroundcolor = white
    color = black
  ;

  /* Tables */
  class table /
    frame = box
    rules = groups
    borderwidth = 1px
    borderstyle = solid
```

⁴ Unfortunately, SAS 9.2 tables use the CELLPADDING= concept and do not support PADDING= on the cells themselves, so this change will work only in SAS 9.3.

Unveiling the Power of Cascading Style Sheets (CSS) in ODS, continued

```

        bordercolor = black
        borderspacing = 0
        bordercollapse = collapse
        /** REMOVE CELLPADDING -- cellpadding = 5px ***/
    ;
    class data, header, rowheader /
        fontfamily = '"Lucida Grande", Arial, sans-serif'
        fontsize = 11pt
        backgroundcolor = #f0f0f0
        color = black
        padding = 5px /** ADD PADDING ***/
    ;
    class header, rowheader /
        fontweight = bold
        textalign = left
        fontsize = 12pt
        backgroundcolor = #d0d0d0
    ;

    /* System Title and Footers */
    class systemtitle, systemfooter /
        fontfamily = '"Lucida Grande", Arial, sans-serif'
    ;

    /* Page number and date (for printer) */
    class pageno, bodydate /
        fontfamily = '"Lucida Grande", Arial, sans-serif'
    ;
end;
run;

```

The output from this style will look just like the output from the very first example. In addition, if you use the CSS code generated in the ODS HTML output on the CSSSTYLE= option, the output will also look nearly the same. The only remaining difference lies with the FRAME= and RULES= attributes that control borders on the table. There are no comparable attributes in CSS; all border control is done explicitly on a cell-by-cell basis to get the same effects as those attributes⁵.

I mentioned in the first section that the idea for specifying multiple style names in the same element came from CSS. This is not evident in the CSS generated by ODS HTML, but we can reformat the above CSS in the following way to simplify it and make it look more like our PROC TEMPLATE style definition.

```

/* Body */
.body {
    background-color: #FFFFFF;
    color: #000000;
}

/* Tables */
.table {
    border: 1px solid #000000;
    border-collapse: collapse;
    border-spacing: 0px;
}
.data, .header, .rowheader {
    background-color: #F0F0F0;
    color: #000000;
    font-family: "Lucida Grande", Arial, sans-serif;
    padding: 5px;
    font-size: 11pt;
}

```

⁵ Theoretically, you should be able to put borders on the THEAD, TBODY, TFOOT, COL, and COLGROUP elements in HTML to get the same effect, but no Web browser currently supports this.

Unveiling the Power of Cascading Style Sheets (CSS) in ODS, continued

```
.header, .rowheader {
  background-color: #D0D0D0;
  font-size: 12pt;
  font-weight: bold;
  text-align: left;
}

/* System Title and Footers */
.systemfooter, .systemtitle {
  font-family: "Lucida Grande", Arial, sans-serif;
  font-size: 13pt;
  font-weight: bold;
}

/* Page number and date (for printer) */
.pageno, .bodydate {
  font-family: "Lucida Grande", Arial, sans-serif;
}
```

This looks much closer to the original PROC TEMPLATE style example. We will use this as the starting point for all subsequent examples.

MEDIA TYPES

One feature that CSS styles give over PROC TEMPLATE styles is that you can define different behaviors for different output types. For example, you might want a style definition to look pretty much the same for HTML and PDF, but you want the HTML output to have color while the PDF remains gray-scale. To do this, you could use media types. Media types within a CSS file tell ODS which sections of the style code apply to which types of output. The basic syntax is as follows:

```
/* Styles for all output types */

@media screen {
  /* Styles for screen output (default) */
}

@media print {
  /* Styles for print */
}
```

The default media type is SCREEN. Any style elements defined without a media type or with a media type of SCREEN will be applied to all output. Other media types must be selected using the CSSSTYLE= option. The media type names in ODS are arbitrary. You can make up any names you wish to suit your needs. The syntax for selecting a media type on the CSSSTYLE= option is as follows:

```
ods output-type cssstyle='filename.css'(media-type-1 <... media-type-n>);
```

The media type names are specified in parentheses after the CSS filename. You can specify multiple media types separated by spaces inside the parentheses if you want to use more than one.

Let's take our last CSS example and make the HTML more colorful than the print version.

```
/* Import previous example style sheet */
@import 'base.css';

/* Add color to body and table cells */
.body { background-color: #c0c0ff; }
.data { background-color: #f0f0ff; }
.header, .rowheader { background-color: #e0e0ff; }

/* Styles to override for print */
@media print {
  .body, .data { background-color: white; }
  .header, .rowheader { background-color: #D0D0D0; }
```


Unveiling the Power of Cascading Style Sheets (CSS) in ODS, continued

```
}
```

The code to generate the output will look like this. Notice the PRINT media type specified at the end of the ODS PDF CSSSTYLE= option. This selects the print media type sections of the CSS in addition to the global style elements.

```
ods html cssstyle='ex3.css';
ods pdf  cssstyle='ex3.css'(print);

proc print data=sashelp.class;
run;

ods _all_ close;
```

The output from HTML and PDF now looks different because we applied different colors based on the CSS media type.

HTML						PDF					
The SAS System						The SAS System					
Obs	Name	Sex	Age	Height	Weight	Obs	Name	Sex	Age	Height	Weight
1	Alfred	M	14	69.0	112.5	1	Alfred	M	14	69.0	112.5
2	Alice	F	13	56.5	84.0	2	Alice	F	13	56.5	84.0
3	Barbara	F	13	65.3	98.0	3	Barbara	F	13	65.3	98.0
4	Carol	F	14	62.8	102.5	4	Carol	F	14	62.8	102.5
5	Henry	M	14	63.5	102.5	5	Henry	M	14	63.5	102.5
6	James	M	12	57.3	83.0	6	James	M	12	57.3	83.0
7	Jane	F	12	59.8	84.5	7	Jane	F	12	59.8	84.5
8	Janet	F	15	62.5	112.5	8	Janet	F	15	62.5	112.5
9	Jeffrey	M	13	62.5	84.0	9	Jeffrey	M	13	62.5	84.0
10	John	M	12	59.0	99.5	10	John	M	12	59.0	99.5
11	Joyce	F	11	51.3	50.5	11	Joyce	F	11	51.3	50.5
12	Judy	F	14	64.3	90.0	12	Judy	F	14	64.3	90.0
13	Louise	F	12	56.3	77.0	13	Louise	F	12	56.3	77.0
14	Mary	F	15	66.5	112.0	14	Mary	F	15	66.5	112.0
15	Philip	M	16	72.0	150.0	15	Philip	M	16	72.0	150.0
16	Robert	M	12	64.8	128.0	16	Robert	M	12	64.8	128.0
17	Ronald	M	15	67.0	133.0	17	Ronald	M	15	67.0	133.0
18	Thomas	M	11	57.5	85.0	18	Thomas	M	11	57.5	85.0
19	William	M	15	66.5	112.0	19	William	M	15	66.5	112.0

Notice in the CSS code above that we did not have to change much to make the two output types look different. We only had to override the attributes that we wanted to look different. All of the other attribute values were simply applied normally using the cascading inheritance effect.

Of course, you can apply differences to more than just colors. Any attribute can be changed within the media section. Not only that, you do not even have to use the same CSS element selectors in the new media type section. We will discuss different types of CSS selectors later.

THE @IMPORT STATEMENT

You might have noticed that we just used the @import statement in the last example. It gives you an easy way to include the contents of another style sheet before making further changes to the styles. The syntax is very simple.

Unveiling the Power of Cascading Style Sheets (CSS) in ODS, continued

```
@import 'file-path';
```

The style elements from that file will be imported just as if they had been typed in the file being loaded.

CSS SELECTORS

So far the only CSS selector we have seen in our code was the class selector. A CSS class selector is indicated by prefixing the name of the class with a period (for example, .data, .header). This corresponds to the HTML CLASS= attribute, which can have one or more class names specified to select various style elements from the CSS file. However, CSS adds many more ways of selecting style elements for use in the document including element name selectors, attribute selectors, and pseudo-class selectors. These extended selector types are new to SAS 9.3 and are still considered preproduction.

ELEMENT NAME SELECTORS

Element name selectors will select elements based on their name. In HTML, this corresponds to the name of the tag. This is table for tables, tr for table rows, td for table data cells, th for table header cells, body for the body of the document, and so on. Since the overall model of reports in ODS was based on HTML, we also use the HTML nomenclature for element names in CSS selectors.

Rather than use class selectors like all of the previous examples, let's write a CSS file that uses only element selectors as described above.

```
body {
    background-color: #FFFFFF;
    color: #000000;
}
table {
    border: 1px solid #000000;
    border-collapse: collapse;
    border-spacing: 0px;
}
td {
    background-color: #F0F0F0;
    color: #000000;
    font-family: "Lucida Grande", Arial, sans-serif;
    padding: 5px;
    font-size: 11pt;
}
```

Unveiling the Power of Cascading Style Sheets (CSS) in ODS, continued

HTML						PDF					
The SAS System						The SAS System					
Obs	Name	Sex	Age	Height	Weight	Obs	Name	Sex	Age	Height	Weight
1	Alfred	M	14	69.0	112.5	1	Alfred	M	14	69.0	112.5
2	Alice	F	13	56.5	84.0	2	Alice	F	13	56.5	84.0
3	Barbara	F	13	65.3	98.0	3	Barbara	F	13	65.3	98.0
4	Carol	F	14	62.8	102.5	4	Carol	F	14	62.8	102.5
5	Henry	M	14	63.5	102.5	5	Henry	M	14	63.5	102.5
6	James	M	12	57.3	83.0	6	James	M	12	57.3	83.0
7	Jane	F	12	59.8	84.5	7	Jane	F	12	59.8	84.5
8	Janet	F	15	62.5	112.5	8	Janet	F	15	62.5	112.5
9	Jeffrey	M	13	62.5	84.0	9	Jeffrey	M	13	62.5	84.0
10	John	M	12	59.0	99.5	10	John	M	12	59.0	99.5
11	Joyce	F	11	51.3	50.5	11	Joyce	F	11	51.3	50.5
12	Judy	F	14	64.3	90.0	12	Judy	F	14	64.3	90.0
13	Louise	F	12	56.3	77.0	13	Louise	F	12	56.3	77.0
14	Mary	F	15	66.5	112.0	14	Mary	F	15	66.5	112.0
15	Philip	M	16	72.0	150.0	15	Philip	M	16	72.0	150.0
16	Robert	M	12	64.8	128.0	16	Robert	M	12	64.8	128.0
17	Ronald	M	15	67.0	133.0	17	Ronald	M	15	67.0	133.0
18	Thomas	M	11	57.5	85.0	18	Thomas	M	11	57.5	85.0
19	William	M	15	66.5	112.0	19	William	M	15	66.5	112.0

This output might not look exactly like you had expected. The system title is colored just like the table and the header elements are the same color as the data cells. The reason that system titles are the same color as tables is that system titles are implemented as tables in HTML. Because there are other components of ODS reports that also use the table element name, it is usually best to use the class selector (that is, .table) when you want a data table.

The table cells are all the same color because ODS does not have a true concept of a header cell; it simply puts a different style on "header" cells to make them look different. However, there is a way to address column headings using a more complex CSS selector.

Just like in HTML, ODS has the concept of a header section on a table. The name of this element is THEAD⁶. You can create a CSS selector that specifies that all TDs within a THEAD should look a certain way simply by combining the two selectors with a space.

```
thead td {
  background-color: #d0d0d0;
  border-width: 0px;
  border-bottom: 1px solid black;
}
```

If we add the above style element to our previous CSS code, we get the following output.

⁶ You can address the body section of a table with TBODY and the footer section of a table with TFOOT.

Unveiling the Power of Cascading Style Sheets (CSS) in ODS, continued

HTML						PDF					
The SAS System						The SAS System					
Obs	Name	Sex	Age	Height	Weight	Obs	Name	Sex	Age	Height	Weight
1	Alfred	M	14	69.0	112.5	1	Alfred	M	14	69.0	112.5
2	Alice	F	13	56.5	84.0	2	Alice	F	13	56.5	84.0
3	Barbara	F	13	65.3	98.0	3	Barbara	F	13	65.3	98.0
4	Carol	F	14	62.8	102.5	4	Carol	F	14	62.8	102.5
5	Henry	M	14	63.5	102.5	5	Henry	M	14	63.5	102.5
6	James	M	12	57.3	83.0	6	James	M	12	57.3	83.0
7	Jane	F	12	59.8	84.5	7	Jane	F	12	59.8	84.5
8	Janet	F	15	62.5	112.5	8	Janet	F	15	62.5	112.5
9	Jeffrey	M	13	62.5	84.0	9	Jeffrey	M	13	62.5	84.0
10	John	M	12	59.0	99.5	10	John	M	12	59.0	99.5
11	Joyce	F	11	51.3	50.5	11	Joyce	F	11	51.3	50.5
12	Judy	F	14	64.3	90.0	12	Judy	F	14	64.3	90.0
13	Louise	F	12	56.3	77.0	13	Louise	F	12	56.3	77.0
14	Mary	F	15	66.5	112.0	14	Mary	F	15	66.5	112.0
15	Philip	M	16	72.0	150.0	15	Philip	M	16	72.0	150.0
16	Robert	M	12	64.8	128.0	16	Robert	M	12	64.8	128.0
17	Ronald	M	15	67.0	133.0	17	Ronald	M	15	67.0	133.0
18	Thomas	M	11	57.5	85.0	18	Thomas	M	11	57.5	85.0
19	William	M	15	66.5	112.0	19	William	M	15	66.5	112.0

You can see in the output above that the cells in the header of the table are a darker gray and they also have a bottom border on them.

SELECTOR CHAINS

We have seen one way of combining selectors simply by putting a space between them, which results in a selector that means the element on the right must be a descendant of the element on the left. This is the most common type of combination in CSS and it can be done using any two selectors, not just element names. For example, if you wanted to only match TD elements within tables that have the class name TABLE, you would use:

```
.table td { ... }
```

If you only wanted the elements within the head of a table with class TABLE, you would use:

```
.table thead td { ... }
```

For cells that exist only within the body of a table with class TABLE, you would use:

```
.table tbody td { ... }
```

Each component of a selector chain can include more than a single selector as well. For example, if you wanted to select only table cells that had the class ROWHEADER, you could use:

```
td.rowheader { ... }
```

Unveiling the Power of Cascading Style Sheets (CSS) in ODS, continued

A more restrictive type of descendant combination is to force the descendant to be a direct descendant (that is, child). Using a space between element selectors means that the ancestor element can be anywhere above the current element in the tree, but if you use > between the element selectors, the descendant must be a child of the element on the left side. This will be more useful in the future with more complex page layouts. Here is an example of a direct descendant chain where each node must be a child of the node on the left side of the operator.

```
.table > tbody > tr > td { ... }
```

The direct descendant operator can also be used to speed up selector chain performance since it will stop looking for the parent node in one step rather than continue to search up the tree in the general descendant combination.

In addition to descendant combinations, it is also possible to select a node based on the previous sibling element. This is done by putting a + between the two element selectors. The following code will select a TD element that has an immediate previous TD sibling with a ROWHEADER class⁷.

```
td.rowheader + td { ... }
```

While element name and class selectors are probably the most common types of CSS selectors you will use, there are more advanced ways of selecting styles. These are discussed in the following sections.

ATTRIBUTE SELECTORS

There are a few attributes available within CSS selectors that allow you to dynamically select styles. Some of the commonly supplied attributes are ID, CLASS, ROWSPAN, COLSPAN, TYPE, and UNFORMATTEDTYPE. Not all of these attributes are supplied by all procedures, so you will have to try them out to see if they will work for you⁸. It is inevitable that more will be added in the future and they will be more consistent across procedures and output types. The general syntax for attribute selectors is as follows:

```
[ attr-name = attr-value ]
```

Attribute selectors can be useful for selecting styles based on the data type of the cell. This is done with the type attribute. The type attribute has two possible values: NUM and CHAR. The code below shows how to apply styles based on data type.

```
/* styles for numeric cells */
tbody td[type=num] {
    font-family: "Courier New", fixed;
    text-decoration: underline;
}
/* styles for character cells */
tbody td[type=char] {
    background-color: yellow;
}
```

⁷ In SAS 9.3, only one immediate previous sibling combination is allowed. You cannot create a long chain.

⁸ Determining what attributes are available is discussed in the section "Finding Element Names and Contexts."

Unveiling the Power of Cascading Style Sheets (CSS) in ODS, continued

HTML						PDF					
The SAS System						The SAS System					
Obs	Name	Sex	Age	Height	Weight	Obs	Name	Sex	Age	Height	Weight
1	Alfred	M	14	69.0	112.5	1	Alfred	M	14	69.0	112.5
2	Alice	F	13	56.5	84.0	2	Alice	F	13	56.5	84.0
3	Barbara	F	13	65.3	98.0	3	Barbara	F	13	65.3	98.0
4	Carol	F	14	62.8	102.5	4	Carol	F	14	62.8	102.5
5	Henry	M	14	63.5	102.5	5	Henry	M	14	63.5	102.5
6	James	M	12	57.3	83.0	6	James	M	12	57.3	83.0
7	Jane	F	12	59.8	84.5	7	Jane	F	12	59.8	84.5
8	Janet	F	15	62.5	112.5	8	Janet	F	15	62.5	112.5
9	Jeffrey	M	13	62.5	84.0	9	Jeffrey	M	13	62.5	84.0
10	John	M	12	59.0	99.5	10	John	M	12	59.0	99.5
11	Joyce	F	11	51.3	50.5	11	Joyce	F	11	51.3	50.5
12	Judy	F	14	64.3	90.0	12	Judy	F	14	64.3	90.0
13	Louise	F	12	56.3	77.0	13	Louise	F	12	56.3	77.0
14	Mary	F	15	66.5	112.0	14	Mary	F	15	66.5	112.0
15	Philip	M	16	72.0	150.0	15	Philip	M	16	72.0	150.0
16	Robert	M	12	64.8	128.0	16	Robert	M	12	64.8	128.0
17	Ronald	M	15	67.0	133.0	17	Ronald	M	15	67.0	133.0
18	Thomas	M	11	57.5	85.0	18	Thomas	M	11	57.5	85.0
19	William	M	15	66.5	112.0	19	William	M	15	66.5	112.0

The code above also uses the TD element name selector in addition to the attribute selector. This is mainly for performance reasons. Attribute selectors are typically slower than class and element name selectors, so it is a good idea to narrow the focus of where attribute selectors are applied. Since we know that only table cells will have the data type attribute set, we can narrow the focus of the selector using the TD element name. We also narrowed the focus down to just the TBODY section of the table so that only the data cells, not the header cells, would be affected.

Another attribute that we can use to subset styles is CLASS. While you can use the built-in syntax for classes (that is, .classname), attribute selectors give you more power than just an exact match. Attribute selectors can be used to match part of an attribute value rather than the whole thing. This is done by using one of the following operators instead of the equal sign.

Operator	Description
~=	Specified value matches any whole word in an attribute value that consists of a space separated list of words
^=	Attribute value starts with a specified value
\$=	Attribute value ends with a specified value
*=	Specified value is found anywhere in the attribute value
=	Attribute value is either the specified value or starts with the specified value and is immediately followed by a hyphen

Unveiling the Power of Cascading Style Sheets (CSS) in ODS, continued

Let's say that we want to match any table cell that has a class name that ends with "header". This will capture both rowheader and header cells in a table. The syntax for this would look like this:

```
td[class$=header] {
    background-color: red;
    color: white;
}
```

HTML						PDF					
The SAS System						The SAS System					
Obs	Name	Sex	Age	Height	Weight	Obs	Name	Sex	Age	Height	Weight
1	Alfred	M	14	69.0	112.5	1	Alfred	M	14	69.0	112.5
2	Alice	F	13	56.5	84.0	2	Alice	F	13	56.5	84.0
3	Barbara	F	13	65.3	98.0	3	Barbara	F	13	65.3	98.0
4	Carol	F	14	62.8	102.5	4	Carol	F	14	62.8	102.5
5	Henry	M	14	63.5	102.5	5	Henry	M	14	63.5	102.5
6	James	M	12	57.3	83.0	6	James	M	12	57.3	83.0
7	Jane	F	12	59.8	84.5	7	Jane	F	12	59.8	84.5
8	Janet	F	15	62.5	112.5	8	Janet	F	15	62.5	112.5
9	Jeffrey	M	13	62.5	84.0	9	Jeffrey	M	13	62.5	84.0
10	John	M	12	59.0	99.5	10	John	M	12	59.0	99.5
11	Joyce	F	11	51.3	50.5	11	Joyce	F	11	51.3	50.5
12	Judy	F	14	64.3	90.0	12	Judy	F	14	64.3	90.0
13	Louise	F	12	56.3	77.0	13	Louise	F	12	56.3	77.0
14	Mary	F	15	66.5	112.0	14	Mary	F	15	66.5	112.0
15	Philip	M	16	72.0	150.0	15	Philip	M	16	72.0	150.0
16	Robert	M	12	64.8	128.0	16	Robert	M	12	64.8	128.0
17	Ronald	M	15	67.0	133.0	17	Ronald	M	15	67.0	133.0
18	Thomas	M	11	57.5	85.0	18	Thomas	M	11	57.5	85.0
19	William	M	15	66.5	112.0	19	William	M	15	66.5	112.0

PSEUDO-CLASS SELECTORS

Pseudo-class selectors give you access to information about the document that is not available from element names, attribute values, or content. They generally surface information about the overall structure of the document or user interactions with the document. The pseudo-selectors currently available in ODS are shown in the following table.

Pseudo-Class	Description
:root	Is the element the root of the document?
:nth-child($an+b$)	Divides the element's children into a groups and selects the b th element.
:nth-of-type($an+b$)	Divides the element's children of the same type (that is, same element name) into a groups and selects the b th element of each group.
:first-child	Is this the first child of the parent element?
:first-of-type	Is this the first child of this type (that is, same element name) of the parent element?
:not(...)	Evaluates the selector within parentheses and negates the result.

Unveiling the Power of Cascading Style Sheets (CSS) in ODS, continued

Probably the most commonly used pseudo-class selector from this group is `nth-child`. It allows you to select elements based on their position within the parent element's context. Why is this so great you might ask? This allows you to do things like alternate row colors within a table without having to do any fancy procedure code or table template writing. In fact, alternating colors is so common that the usual `an+b` argument has two shortcuts that can be used: `EVEN` and `ODD`. Here is a simple example of CSS code that will alternate the row colors in a table⁹.

```
.table tbody tr:nth-child(even) td {
  background-color: white;
  color: black;
}
.table tbody tr:nth-child(odd) td {
  background-color: black;
  color: white;
}
```

HTML						PDF					
The SAS System						The SAS System					
Obs	Name	Sex	Age	Height	Weight	Obs	Name	Sex	Age	Height	Weight
1	Alfred	M	14	69.0	112.5	1	Alfred	M	14	69.0	112.5
2	Alice	F	13	56.5	84.0	2	Alice	F	13	56.5	84.0
3	Barbara	F	13	65.3	98.0	3	Barbara	F	13	65.3	98.0
4	Carol	F	14	62.8	102.5	4	Carol	F	14	62.8	102.5
5	Henry	M	14	63.5	102.5	5	Henry	M	14	63.5	102.5
6	James	M	12	57.3	83.0	6	James	M	12	57.3	83.0
7	Jane	F	12	59.8	84.5	7	Jane	F	12	59.8	84.5
8	Janet	F	15	62.5	112.5	8	Janet	F	15	62.5	112.5
9	Jeffrey	M	13	62.5	84.0	9	Jeffrey	M	13	62.5	84.0
10	John	M	12	59.0	99.5	10	John	M	12	59.0	99.5
11	Joyce	F	11	51.3	50.5	11	Joyce	F	11	51.3	50.5
12	Judy	F	14	64.3	90.0	12	Judy	F	14	64.3	90.0
13	Louise	F	12	56.3	77.0	13	Louise	F	12	56.3	77.0
14	Mary	F	15	66.5	112.0	14	Mary	F	15	66.5	112.0
15	Philip	M	16	72.0	150.0	15	Philip	M	16	72.0	150.0
16	Robert	M	12	64.8	128.0	16	Robert	M	12	64.8	128.0
17	Ronald	M	15	67.0	133.0	17	Ronald	M	15	67.0	133.0
18	Thomas	M	11	57.5	85.0	18	Thomas	M	11	57.5	85.0
19	William	M	15	66.5	112.0	19	William	M	15	66.5	112.0

In this case, we had to apply the `nth-child` pseudo-class selector to the `TR` (table row) element, but we also have to use `TD` at the end of the selector since some output types require all style attributes to be at the cell level.

⁹ You might have noticed that HTML starts with white and PDF starts with black. There is an issue in HTML (actually, all tagsets) that causes them to start at the incorrect numbering. This should be fixed in a future version of SAS.

Unveiling the Power of Cascading Style Sheets (CSS) in ODS, continued

Just as with attribute selectors, it is a good idea to narrow the focus of your pseudo-selectors down to just the elements to which they apply. In this case, we are subsetting using a `.table` class selector to get just data tables, then subsetting that by the `TBODY` element to apply only to the body of the table.

Of course, you can do different types of alternating as well using the more formal `an+b` formula. Here is an example that alternates colors every three rows¹⁰.

```
.table tbody tr:nth-child(6n+1) td,
.table tbody tr:nth-child(6n+2) td,
.table tbody tr:nth-child(6n+3) td {
  background-color: white;
  color: black;
}
.table tbody tr:nth-child(6n+4) td,
.table tbody tr:nth-child(6n+5) td,
.table tbody tr:nth-child(6n+6) td {
  background-color: black; color: white;
}
```

HTML						PDF					
The SAS System						The SAS System					
Obs	Name	Sex	Age	Height	Weight	Obs	Name	Sex	Age	Height	Weight
1	Alfred	M	14	69.0	112.5	1	Alfred	M	14	69.0	112.5
2	Alice	F	13	56.5	84.0	2	Alice	F	13	56.5	84.0
3	Barbara	F	13	65.3	98.0	3	Barbara	F	13	65.3	98.0
4	Carol	F	14	62.8	102.5	4	Carol	F	14	62.8	102.5
5	Henry	M	14	63.5	102.5	5	Henry	M	14	63.5	102.5
6	James	M	12	57.3	83.0	6	James	M	12	57.3	83.0
7	Jane	F	12	59.8	84.5	7	Jane	F	12	59.8	84.5
8	Janet	F	15	62.5	112.5	8	Janet	F	15	62.5	112.5
9	Jeffrey	M	13	62.5	84.0	9	Jeffrey	M	13	62.5	84.0
10	John	M	12	59.0	99.5	10	John	M	12	59.0	99.5
11	Joyce	F	11	51.3	50.5	11	Joyce	F	11	51.3	50.5
12	Judy	F	14	64.3	90.0	12	Judy	F	14	64.3	90.0
13	Louise	F	12	56.3	77.0	13	Louise	F	12	56.3	77.0
14	Mary	F	15	66.5	112.0	14	Mary	F	15	66.5	112.0
15	Philip	M	16	72.0	150.0	15	Philip	M	16	72.0	150.0
16	Robert	M	12	64.8	128.0	16	Robert	M	12	64.8	128.0
17	Ronald	M	15	67.0	133.0	17	Ronald	M	15	67.0	133.0
18	Thomas	M	11	57.5	85.0	18	Thomas	M	11	57.5	85.0
19	William	M	15	66.5	112.0	19	William	M	15	66.5	112.0

¹⁰ The first batch of rows in HTML only contain two rows. This is due to the same issue as mentioned in the previous example.

Unveiling the Power of Cascading Style Sheets (CSS) in ODS, continued

One use for the nth-child pseudo-class selector has nothing to do with alternating colors; you can specify different styles for each cell in a multi-part system title. A TITLE statement such as the following will create a three part system title where each of the parts has a class of SYSTEMTITLE.

```
title j=l 'Left' j=c 'Center' j=r 'Right';
```

You can address each of the parts of the system title using the nth-child pseudo-class selector as follows:

```
.systemtitle:nth-child(1) {
  background-color: blue;
  color: white;
  font-style: italic;
  font-family: times;
}
.systemtitle:nth-child(2) {
  background-color: red;
  color: white;
  font-size: 20pt;
  font-family: impact;
  border: 1px solid black;
}
.systemtitle:nth-child(3) {
  background-color: #f0f0f0;
  color: black;
  font-style: italic;
  font-family: times;
}
```

HTML

Left

Center

Right

Obs	Name	Sex	Age	Height	Weight
1	Alfred	M	14	69.0	112.5
2	Alice	F	13	56.5	84.0
3	Barbara	F	13	65.3	98.0
4	Carol	F	14	62.8	102.5
5	Henry	M	14	63.5	102.5
6	James	M	12	57.3	83.0
7	Jane	F	12	59.8	84.5
8	Janet	F	15	62.5	112.5
9	Jeffrey	M	13	62.5	84.0
10	John	M	12	59.0	99.5
11	Joyce	F	11	51.3	50.5
12	Judy	F	14	64.3	90.0
13	Louise	F	12	56.3	77.0
14	Mary	F	15	66.5	112.0
15	Philip	M	16	72.0	150.0
16	Robert	M	12	64.8	128.0
17	Ronald	M	15	67.0	133.0
18	Thomas	M	11	57.5	85.0
19	William	M	15	66.5	112.0

PDF

Left

Center

Right

Obs	Name	Sex	Age	Height	Weight
1	Alfred	M	14	69.0	112.5
2	Alice	F	13	56.5	84.0
3	Barbara	F	13	65.3	98.0
4	Carol	F	14	62.8	102.5
5	Henry	M	14	63.5	102.5
6	James	M	12	57.3	83.0
7	Jane	F	12	59.8	84.5
8	Janet	F	15	62.5	112.5
9	Jeffrey	M	13	62.5	84.0
10	John	M	12	59.0	99.5
11	Joyce	F	11	51.3	50.5
12	Judy	F	14	64.3	90.0
13	Louise	F	12	56.3	77.0
14	Mary	F	15	66.5	112.0
15	Phillp	M	16	72.0	150.0
16	Robert	M	12	64.8	128.0
17	Ronald	M	15	67.0	133.0
18	Thomas	M	11	57.5	85.0
19	William	M	15	66.5	112.0

Another interesting pseudo-class selector is the :not selector. It allows you to specify a selector pattern that you do not want matched, rather than one you do want matched. Let's say that you want to match every table cell that has a class name that does not start with "row". You could do that with the following selector.

```
.table tbody td:not( [class^=row] ) {
  background-color: green;
  color: white;
```

Unveiling the Power of Cascading Style Sheets (CSS) in ODS, continued

```
}
```

HTML						PDF					
The SAS System						The SAS System					
Obs	Name	Sex	Age	Height	Weight	Obs	Name	Sex	Age	Height	Weight
1	Alfred	M	14	69.0	112.5	1	Alfred	M	14	69.0	112.5
2	Alice	F	13	56.5	84.0	2	Alice	F	13	56.5	84.0
3	Barbara	F	13	65.3	98.0	3	Barbara	F	13	65.3	98.0
4	Carol	F	14	62.8	102.5	4	Carol	F	14	62.8	102.5
5	Henry	M	14	63.5	102.5	5	Henry	M	14	63.5	102.5
6	James	M	12	57.3	83.0	6	James	M	12	57.3	83.0
7	Jane	F	12	59.8	84.5	7	Jane	F	12	59.8	84.5
8	Janet	F	15	62.5	112.5	8	Janet	F	15	62.5	112.5
9	Jeffrey	M	13	62.5	84.0	9	Jeffrey	M	13	62.5	84.0
10	John	M	12	59.0	99.5	10	John	M	12	59.0	99.5
11	Joyce	F	11	51.3	50.5	11	Joyce	F	11	51.3	50.5
12	Judy	F	14	64.3	90.0	12	Judy	F	14	64.3	90.0
13	Louise	F	12	56.3	77.0	13	Louise	F	12	56.3	77.0
14	Mary	F	15	66.5	112.0	14	Mary	F	15	66.5	112.0
15	Philip	M	16	72.0	150.0	15	Philip	M	16	72.0	150.0
16	Robert	M	12	64.8	128.0	16	Robert	M	12	64.8	128.0
17	Ronald	M	15	67.0	133.0	17	Ronald	M	15	67.0	133.0
18	Thomas	M	11	57.5	85.0	18	Thomas	M	11	57.5	85.0
19	William	M	15	66.5	112.0	19	William	M	15	66.5	112.0

As you can see, pseudo-class selectors offer a great number of possible enhancements to your reports.

THE GORY DETAILS

So far in this paper, we have tried to provide guidance in applying CSS styles to your reports and have given you some tips on how to address common elements. However, there are some important details that you should know about how style elements are applied and how to know what elements are available.

FINDING ELEMENT NAMES AND CONTEXTS

There is a logging tool built into ODS to help you determine the context of elements within a report in order to determine what CSS context to use to address them. To turn this feature on, use the following statement.

```
ods trace dom;
```

The acronym DOM stands for "document object model". That simply means the structure of the document including element names and attributes.

When this feature is turned on, markup will be printed to the log that is very XML/HTML-like. Here is a sample of the output you will see. This sample is of a fairly common table row in HTML.

```
<html4:tr > ROW_START
  <html4:td class="RowHeader" rowspan="1" type="num" unformatted-type="num"> CELL_START
```

Unveiling the Power of Cascading Style Sheets (CSS) in ODS, continued

```

</html4:td> CELL_START
<html4:td class="Data" rowspan="1" type="char" unformatted-type="char"> CELL_START
</html4:td> CELL_START
<html4:td class="Data" rowspan="1" type="char" unformatted-type="char"> CELL_START
</html4:td> CELL_START
<html4:td class="Data" rowspan="1" type="char" unformatted-type="char"> CELL_START
</html4:td> CELL_START
<html4:td class="Data" rowspan="1" type="char" unformatted-type="char"> CELL_START
</html4:td> CELL_START
</html4:tr> ROW_START

```

The element names are prefixed with "html:". This indicates which ODS output type generated the messages. The element name comes next. In this case, we see only TR and TD elements. The rest of the element tag contains the attributes. Here we see CLASS, ROWSPAN, TYPE, and UNFORMATTED-TYPE- attributes. The term at the end of the line indicates which method on the document was called to generate that piece of the report. It can be ignored.

Using the output from ODS TRACE DOM, we have all of the information about the structure of the document that can be used to construct CSS selectors. Most of the time it will be fairly easy to infer what the structure is simply by looking at the source of the HTML code, but some attributes such as TYPE and UNFORMATTED-TYPE would not be evident.

SPECIFICITY

We have not mentioned anything about the precedence of style elements within a CSS file. So far, we have said only that style attributes for the same selector specified later within a file will override the same attributes set earlier in the file. This was never an issue with PROC TEMPLATE styles because only one style element could be applied to any part of a report. This is not the case with the CSS style engine.

The precedence of elements within a CSS file is determined by the CSS selector's "specificity." Calculating the specificity of a selector is fairly easy. Count the number of ID selectors and set that to A. Count the number of class, attribute, and pseudo-class selectors and set that value to B¹¹. Count the number of element type selectors and set that value to C. Then simply concatenate those three numbers as A-B-C. The resulting number is the specificity of that CSS selector. The higher the specificity, the higher the precedence of the style attributes in that style element. Here are some example specificities.

Selector	Variables	Specificity
td	a=0 b=0 c=0	1
td.rowheader	a=0 b=1 c=1	11
tbody td[type=char]	a=0 b=1 c=2	12
.systemtitle:nth-child(1)	a=0 b=2 c=0	20
.table tbody td:not([class^=row])	a=0 b=2 c=2	22

UNIVERSAL SELECTOR

There is one selector that does not require any knowledge at all about the element you want to match. It is called the "universal selector." This selector is essentially just a wildcard. It matches any element name. This selector is represented by an asterisk (*). The universal selector can be used to apply style attributes to every element in a document as follows.

```

*{
    /* global style attributes */
}

```

Unless you really need it, this is not recommended because it can cause performance issues, and there are very few attributes that would be useful when applied to every element in a document. However, the universal selector is useful for dealing with selector chains. Let's say that you want to match a table row of a data table, but you do not care which section of the table it is in (that is, THEAD, TBODY, TFOOT). In addition, you want to indicate that the

¹¹ While all selectors within a :not pseudo-class are counted, the :not itself isn't.

Unveiling the Power of Cascading Style Sheets (CSS) in ODS, continued

table row is from that table and not within a nested table. You could use the universal selector in combination with the child combinator as follows:

```
.table > * > tr > td { ... }
```

Using the universal selector, we are making sure that each selector in the chain is a child of the previous selector, but the element between .table and tr can be any single element.

EXAMPLE

The following style example uses most of the features discussed in this paper. I will leave it as an exercise for the reader to go through all of the pieces of this code. If you have trouble figuring out what a section does, you might want to apply this CSS file to your output section by section and see what changes from one run to the next. There are some fairly complex selectors in this file, so if you are not familiar with CSS it might be a bit daunting at first. As with anything, with more practice it will become second nature.

```
@import 'base.css';

/* adjust font size and padding for printer */
@media print {
  .table td {
    font-size: 9pt;
    padding: 5pt;
  }
}

/* make changes to system titles */
.systemtitle {
  font-size: small;
  font-family: times;
  font-style: italic;
  color: #a0a0a0;
  background-color: #f0f0f0;
  width: 100%; /* needed for printer */
  padding: 10pt;
  padding-bottom: 30pt;
}

/* make changes to bylines */
.byline {
  border-width: 0px;
  border-bottom: 2px solid #a0a0a0;
  font-family: 'Lucida Grande', arial, sans-serif;
  font-size: x-large;
  font-weight: bold;
  width: 100%; /* needed for printer */
  padding-bottom: 10pt;
}

/* draw border below table header section */
.table > tbody > tr:nth-child(1) > td {
  border-width: 0px;
  border-top: 1px solid black;
}

/* work around problem in html with child nodes */
@media html {
  .table > tbody > tr:nth-child(2) > td {
    border-width: 0px;
    border-top: 1px solid black;
  }
}

/* draw border to the right of row headers */
```

Unveiling the Power of Cascading Style Sheets (CSS) in ODS, continued

```

.table > tbody > tr:nth-child(1) > td.rowheader,
.table > tbody > tr > td.rowheader {
    border-right: 1px solid black;
    border-bottom: 0px;
}

/* work around problem in html with child nodes */
@media html {
    .table > tbody > tr:nth-child(2) > td.rowheader {
        border-right: 1px solid black;
        border-bottom: 0px;
    }
    .table > tbody > tr:not(:nth-child(2)) > td.rowheader {
        border-top: 0px; border-bottom: 0px;
    }
}

/* alternate even header cell colors */
.table > tbody > tr:nth-child(even) > td:not(.data) {
    background-color: #d0d0d0;
}

/* alternate even data cell colors */
.table > tbody > tr:nth-child(even) > td.data {
    background-color: #f0f0f0;
}

/* alternate odd header cell colors */
.table > tbody > tr:nth-child(odd) > td:not(.data) {
    background-color: #c0c0c0;
}

/* alternate odd data cell colors */
.table > tbody > tr:nth-child(odd) > td.data {
    background-color: #e0e0e0;
}

```

Here is the SAS program that we will use with this style sheet.

```

ods html file='ex13.html' cssstyle='ex13.css'(html);
ods pdf file='ex13.pdf' cssstyle='ex13.css'(print);

proc sort data=sashelp.class out=class; by sex;
run;

proc print data=class; by sex;
run;

ods _all_ close;

```


Unveiling the Power of Cascading Style Sheets (CSS) in ODS, continued

Here is what the HTML and PDF output look like.

HTML

The SAS System

Sex=F

Obs	Name	Age	Height	Weight
1	Alice	13	56.5	84.0
2	Barbara	13	65.3	98.0
3	Carol	14	62.8	102.5
4	Jane	12	59.8	84.5
5	Janet	15	62.5	112.5
6	Joyce	11	51.3	50.5
7	Judy	14	64.3	90.0
8	Louise	12	56.3	77.0
9	Mary	15	66.5	112.0

Sex=M

Obs	Name	Age	Height	Weight
10	Alfred	14	69.0	112.5
11	Henry	14	63.5	102.5
12	James	12	57.3	83.0
13	Jeffrey	13	62.5	84.0
14	John	12	59.0	99.5
15	Philip	16	72.0	150.0
16	Robert	12	64.8	128.0
17	Ronald	15	67.0	133.0
18	Thomas	11	57.5	85.0
19	William	15	66.5	112.0

PDF

The SAS System

Sex=F

Obs	Name	Age	Height	Weight
1	Alice	13	56.5	84.0
2	Barbara	13	65.3	98.0
3	Carol	14	62.8	102.5
4	Jane	12	59.8	84.5
5	Janet	15	62.5	112.5
6	Joyce	11	51.3	50.5
7	Judy	14	64.3	90.0
8	Louise	12	56.3	77.0
9	Mary	15	66.5	112.0

Sex=M

Obs	Name	Age	Height	Weight
10	Alfred	14	69.0	112.5
11	Henry	14	63.5	102.5
12	James	12	57.3	83.0
13	Jeffrey	13	62.5	84.0
14	John	12	59.0	99.5
15	Philip	16	72.0	150.0
16	Robert	12	64.8	128.0
17	Ronald	15	67.0	133.0
18	Thomas	11	57.5	85.0
19	William	15	66.5	112.0

CONCLUSION

As you can see from the capabilities introduced in this paper, the state of ODS styles has taken a huge leap from the past decade. The best part is that CSS is an open specification written by the World Wide Web Consortium (W3C)¹². The goal for the CSS implementation in ODS is to follow that specification as closely as possible. Using a standard style syntax like CSS will allow you to use regular CSS editing tools and resources for your ODS styles. Until the next phase of development arrives, there should be plenty of tips in this paper to keep you busy making beautiful reports.

REFERENCES

"Cascading Style Sheets." World Wide Web Consortium. Available at <http://www.w3.org/Style/CSS/>.

¹² See <http://www.w3.org/Style/CSS/> for more information.

Unveiling the Power of Cascading Style Sheets (CSS) in ODS, continued

CONTACT INFORMATION

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

Name: Kevin D. Smith
Enterprise: SAS
Address: 1 SAS Campus Drive
City, State ZIP: Cary, NC 27513
E-mail: Kevin.Smith@sas.com
Web: <http://www.sas.com/>

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

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