Custom Tagsets to Create Custom HTML
Lori Maiwald, New York State Department of Health

Abstract

Custom tagsets can be developed to overcome html-, tag- or style-restrictions of a large web-site. A web-master may require all HTML documents destined for their site to have specific paired tags orphaned, no frames, no styles and the insertion of additional lines to assist in site-wide substitution of header, meta data and footer. Custom tagsets can relieve the SAS® programmer of the maintenance burden of data/job-specific html coding. If you know your data step and procedures as well as have a clear idea of the html that you require as an end result, you can create your own tagsets to get that result and stop 'put'-ting the HTML in each program you write.

Introduction

Tagsets, one of the many players in the convolution called the ODS (Output Delivery System), are meant to be a customize-able component of SAS® just like styles. A great number of tagsets for HTML, XML, PDF and RTF exist now as part of SAS® 9.1 and were available as experimental code in SAS® 8.2. They, in conjunction with a large number of built-in styles, offer SAS output a range of file formats and appearances far beyond the old output window. Most SAS programmers can restrict their customization to the styles and style attributes. Styles can afford the programmer a vast range of appearance changes and a small amount of choice in the tags used.

When drastic changes to the html are required, whether by web-site restriction or by desire, styles cannot affect enough of the html generated by the current tagsets of phtml, xhtml, html4, and htmlcss. Changes to the most basic of paired codes, such as orphaning <html> with no </html> or omitting <body>...</body> and no styles can only be done in a custom tagset. Such restrictions work to the benefit of a large web-site as a whole since consistency across pages can be enforced and site-wide changes can occur centrally. The restrictions, however, can cause html not to validate and look primitive/raw until put up on the web-server.

How to Start to Customize a Tagset

The easiest way to start a tagset is to inherit one. Choose the one closest to the file format or file content you desire. To get a list of the available tagsets type the following code:

```sas
proc template;
list tagsets;
run;
```

To view/print the contents of the tagset you must go to the Results window, right-click on the word 'Results' and choose Template from the menu. The Template Window will open and list the folders. Click on Sashelp.tmplmst and then click on tags when the folders are listed. Double-click of the tagset of choice and the Template Browser window will open to view or print the tagset.

To start your own tagset, you can begin with the following:

```sas
proc template;
define tagset myweb;
parent=tagsets.html4;
end;
run;
```

Now create an experimental dataset, initialize a few titles/footnotes and try a PROC REPORT or a PROC TABULATE in order to review the source code. (Please note this code requires a subdirectory/folder called 'saswork9' on the c: drive.)

```sas
title1 'My Fruit Experiment';
title2 'Table 1';
footnote1 'my 2005 experiment';
footnote2 'simple example';
```
data myexperiment;
input my_word $ my_number;
cards;
peach 1
pear 2
apple 3
cherry 4
peach 5
pear 6
apple 7
cherry 8
;
run;

filename myexp 'c:\saswork9\myweb_experiment.htm';
ods markup body=myexp tagset=myweb;
proc report data=myexperiment headline nowindows;
column my_word;
column my_number;
break after my_word / ol;
compute after my_word;
line 'subtotal for ' my_word $8. 'is' my_number.sum comma7.2;
endcomp;
define my_word / order order=data ;
define my_number / display sum;
run;
ods markup close;

Your output will be the same as if you performed ODS HTML4 with pages of styles in your heading, the SAS-standard document heading, perfectly paired tags and references to the styles embedded throughout your document. Now you can start changing things.

To Complete the Development of the Customized Tagset

At this point you have a choice of whether to retain the parent statement such that you override each event as you include it in your tagset or you may comment out the parent statement such that the only events that generate html are the ones included in the custom tagset. In this situation, since more events dealing with styles, frames and table of contents were included in the parent style that needed to be removed, it was easier to comment out the parent style and begin by cutting, pasting and editing the desired events into the custom tagset to fulfill the requirements.

You will not correctly identify all the needed events on the first try. Multiple cycles of:
1) putting the parent reference in,
2) re-running,
3) adding back in another event,
4) commenting the parent reference back out, then
5) running again

will be required. Alternatively, you can use the tagsets in SUGI30 paper, “Tagset Spelunking ...” by Eric Gebhart to explore events within procedures. This cycle continues until the HTML (XHTML-strict actually) that is sought is generated. This very basic tagset, with a handful of macro variables so that the required tags may be used in multiple programs, resulted once the experiment’s cycles were completed. It can be used with PROCs PRINT, REPORT and TABULATE.

* set your meta data values;
%let mytitle=My Experiment Program;
%let myhead=My Experiment Program;
%let myupdate=June 2005;
%let myemail=llm07@health.state.ny.us;
%let mydesc=My Experiment Program;
%let mykey=experiment, fruit, tagsets;
%let mylinkt=None;
%let mylinku=/none.htm;
proc template;
define tagset myweb;

Note: macro variable references for information required to be embedded in the heading of the document by the website. This will allow the tagset to be saved to another document so that it can be recalled by an %include statement.
The strip function removes the space padding many PROCs use.

The event doc was crucial, it replaced event doc, top_file, top_title, top_code, doc_head, top_head, doc_meta, top_meta, doc_body and all events to do with javascript. This is a web-site-specific header to be manipulated by the webmaster using the variable values to ensure web document consistency. It is only an example to be replaced by whatever is required at your institution.
Now rerun the PROC REPORT and view the difference in both browser view and source code.
Necessary Variants of the Customized Tagset

This basic tagset functions adequately for single ODS MARKUP calls. Unlike ODS HTML, ODS MARKUP does not support NO_TOP_MATTER nor NO_BOTTOM_MATTER such that you must create 3 slightly different versions of your tagset. The three additional versions of the tagset will enable you to start up the HTML file, add paragraph text, perform other functions, return to the html to add more reporting, leave again to only come back to finish up and close ODS.

The three additional versions are:
1) a ‘begin’ tagset, with the abovementioned basic tagset as parent, with document headings but no closings,
2) an ‘end’ tagset, with the abovementioned basic tagset as parent, with document closings but no headings and
3) a ‘middle’ tagset, with the abovementioned basic tagset as parent, without headings or closings.

These three tagsets are simply constructed by inheritance.

```
proc template;
define tagset myweb_begin;
parent=myweb;
define event doc;
start:
put '<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html lang="en-us">
<head>
<title>'&myhead.' '</title>
<meta name="description" content="&mysdescr." />
<meta name="keywords" content="&mykey." />
<!-- THE FOLLOWING STYLE TAG IS FOR IMPORTING STYLE ONLY -->
<style type="text/css">
<!--
-->
</style>
<!--#set var="SSI_VERSION" value="4" -->
<!--#set var="PAGETITLE" value="&mytitle." -->
<!--#set var="GLOBALNAVIGATION" value="false" -->
<!--#set var="LOCALNAVIGATIONURL" value="false" -->
<!--#set var="SPECIALSECTIONURL" value="false" -->
<!--#set var="RETURNLINKURL" value="&mylinku." -->
<!--#set var="RETURNLINKTEXT" value="&mylinkt." -->
<!--#set var="EMAILADDRESS" value="&myemail." -->
<!--#set var="REVISED_DATE" value="&myupdate." -->
<!--#set var="LANGUAGE" value="en" -->
<!--#set var="FORMATHELP" value="" -->
<!--#include virtual="/include/xhtml/xhtml_header.htm"-->
'</html>
end;

define tagset myweb_end;
parent=myweb;
define event doc;
finish:
put '<!--include virtual="/include/xhtml/xhtml_footer.htm"-->
end;

define tagset myweb_mid;
parent=myweb;
define event doc;
end;
run;
```
Implementation of the Customized Tagsets

You may need to use all three versions of the tagset to construct a single HTML file by the incremental addition of elements like introductory paragraphs and instructions subsequently followed by a table of information that is then followed by more text before the final table. The following code is predicated upon having run all preceding code in this paper.

```
data my_text;
input my_words $82.;
*0000000111111112222222223333333334444444445555555556666666667777777778888;
*234567890112345678901234567890123456789012345678901234567890123456789012;
cards;
<p>I am just inserting a paragraph before getting around to next output procedure. I hope to get around typing so many put statements and quotes. This is to store new paragraphs - the only problem seems to be that the datalines/cards must be the same length. 'Quotes= do NOT throw it off.<p>
</p><p><a href="#jumper1"> jump to second table</a> and new paragraph.</p>
```

For future implementations, the two PROC TEMPLATEs should be saved in a separate file for use in %INCLUDE statements. The macro variable statements would have to be included and initialized in programs for which you wish the tagsets to correctly resolve.

The resultant HTML source code produced by the customized tagsets

The result of running the code will be a single xhtml file, with the following source code:
I am just inserting a paragraph before getting around to next output procedure. I hope to get around typing so many put statements and quotes. This is to store new paragraphs - the only problem seems to be that the datalines/cards must be the same length. "Quotes" do NOT throw it off.

This text is the anchor for the in-document link.
The Resultant Web Page

The code listed above will generate an HTML document that, once put up on the web-server with the centrally-controlled styles, can look like so (otherwise you will see tables without borders, no colors and text in Times Roman):

-->

I am just inserting a paragraph before getting around to next output procedure. I hope to get around typing so many put statements and quotes. This is to store new paragraphs - the only problem seems to be that the datelines/cards must be the same length. "Quotes" do NOT throw it off.

jump to second table and new paragraph.

My Fruit Experiment

Table 1

<table>
<thead>
<tr>
<th>my_word</th>
<th>my_number</th>
</tr>
</thead>
<tbody>
<tr>
<td>peach</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>5</td>
</tr>
<tr>
<td>subtotal for peach is 6.00</td>
<td></td>
</tr>
<tr>
<td>pear</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>6</td>
</tr>
<tr>
<td>subtotal for pear is 8.00</td>
<td></td>
</tr>
<tr>
<td>apple</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>7</td>
</tr>
<tr>
<td>subtotal for apple is 10.00</td>
<td></td>
</tr>
<tr>
<td>cherry</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>8</td>
</tr>
<tr>
<td>subtotal for cherry is 12.00</td>
<td></td>
</tr>
</tbody>
</table>

my 2005 experiment

simple example
Here is the text for the second table.

This text is the anchor for the in-document link

**My Fruit Experiment**

**Table 2**

<table>
<thead>
<tr>
<th>my_number</th>
<th>Sum</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>my_word</strong></td>
<td></td>
</tr>
<tr>
<td>apple</td>
<td>10.00</td>
</tr>
<tr>
<td>cherry</td>
<td>12.00</td>
</tr>
<tr>
<td>peach</td>
<td>6.00</td>
</tr>
<tr>
<td>pear</td>
<td>8.00</td>
</tr>
</tbody>
</table>

*my 2005 experiment*

**simple example**

**Conclusion**

While the code does not cover all events for all procedures, this is a working document that can create html to meet required specifications. This paper offers the example of web-site requirements to illustrate the long term solution to customized html. However, the tagset’s results have been limited in a few ways due to the way SAS® handles some PROCs and the events they encapsulate.

Firstly, the desired HTML included spanned rows for group variables in a PROC REPORT to display data. However only PROC TABULATE spans rows for class variables but does not list/display data where PROC REPORT only spans columns for compute breaks leaving grouping variable value cells empty but for the non-breaking space. Secondly, the desired HTML for a table of one column is `<ul>...<li>...<li>...</ul>` such that you will need to create another tagset in order to produce the HTML from what is ordinarily a table event. Such lists can provide navigation to anchors within tables to meet accessibility standards. Lastly, in order to embed links and anchors which are stored in their own fields within the text of other variables (such as a mailto: link anchored to a person’s name), you must still handle by careful concatenation to a new variable.

Limitations aside, tagsets are a superior way to generate custom html in terms of programming time and long range maintenance. The burden of maintaining large amounts of explicit HTML coding in each and every program is enormous and, now, generally unnecessary.
Acknowledgments

I would like to thank Eric Gebhart of SAS for his assistance with PROC TEMPLATE, styles and tagsets over the past few years. His paper "ODS Markup: The Power of Choice and Change" and his help were pivotal to the successful completion of the task to which this paper's contents applied. Mr. Gebhart's subsequent paper, "Tagset Spelunking and Cartography: Debugging and Exploring Tagsets with Battery-Powered Headlamps", proved invaluable in fully appreciating the events particular to specific procedures.

Gregory Newkirk's paper: "Using ODS, an Easy Approach in Creating HTML Web Pages" was instrumental in the realization that government agencies are regularly hit the hardest with regards to accessibility standards and site-wide consistency standards.

SAS® and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. © indicates USA registration. Other brand and product names are registered trademarks or trademarks of their respective companies.


Contact Information

If you have and questions or comments, write or call:

Lori Maiwald
Program Research Specialist 3
New York State Department of Health
Center for Environmental Health
Flanigan Square, 547 River Street, Room 400
Troy, NY 12180
(518) 402-7714
llm07@health.state.ny.us