Paper 211-2009

Using SAS/Graph®, JavaScript and ODS to Create Interactive HTML Output

James L. Ribble, Columbus Technologies, Inc., Atlanta, Ga.

ABSTRACT

This paper will demonstrate combining tabular data and SAS/Graph output with JavaScript logic to create interactive HTML documents. Examples will be shown of how to customize your web document output using both data steps and ODS. The use of HTML style specifications will also be demonstrated as well as how to define multiple independently controlled graphic layers.

INTRODUCTION

Web documents are a great way of presenting data and SAS provides several convenient ways of creating them. But plain web pages can at times be dull, like looking at slides composed of nothing but text. The viewer must plow through the text to glean the information it contains. Adding graphs and images make web pages visually stimulating and the data easier to comprehend. The real fun begins when a user is presented with a web page that contains controls allowing them to be interactive with the page and select views or subsets of the data to suit their own needs. Web pages such as this invite the user to indulge their curiosity and allow them real-time control of what they are seeing. In this paper I will be presenting some basic concepts and techniques that can be used to create interactive JavaScript-enabled web documents.

THE OBJECTIVE

Our objective will be to create a map of the United States along with a table of data containing clickable checkboxes. These checkboxes will allow the user to select which year(s) of data is to be shown on the map. Our test data will span the years 2001 through 2003. In order to demonstrate the various methods of accomplishing this we will be using both ODS and DATA steps to generate our HTML output.

LAYERING

In our web page we will be using a technique called ‘layering’ in which one graphic image is displayed overtop of another. These layers are essentially images encased in HTML <DIV> (Division) tags. The layers can be turned on and off and controlled in other ways by manipulating the style attributes associated with a given division. Here is a snapshot of HTML showing the layer definitions as they will need to appear in our final output:

```
<div id="US" align="left" valign="top"
     style="position:absolute; top:0; left:35; z-index: 1;">
   <img src="BlankUS.gif">
</div>
<div id="G2001" align="left" valign="top"
     style="position:absolute; top:0; left:35; z-index: 2; visibility:hidden;">
   <img src="year2001.gif">
</div>
<div id="G2002" align="left" valign="top"
     style="position:absolute; top:0; left:35; z-index: 2; visibility:hidden;">
   <img src="year2002.gif">
</div>
<div id="G2003" align="left" valign="top"
     style="position:absolute; top:0; left:35; z-index: 2; visibility:hidden;">
   <img src="year2003.gif">
</div>
```

In the above markup we have defined four layers. One is the blank US outline. The other three are for each of the years to be displayed, i.e. overlaid, on top of the outline. The <IMG> (Image) tags point to the individual graphic images we will be creating with SAS/Graph. The ID attributes on the <DIV> tags assign a name by which the division/layer can be identified. The POSITION, TOP and LEFT style attributes set a fixed location for the graphics so that they will line up properly. The Z-INDEX specification says that the graphics with the higher values should appear over top of those with lower values thus setting the display order of the layers. The VISIBILITY attributes set the three yearly data plots to be ‘hidden’ or invisible when the page is initially displayed. Changing this attribute to ‘visible’ will cause the layer to be displayed. It is this style attribute we will be dynamically manipulating with JavaScript to make the layers appear and disappear.

JAVASCRIPT

In order to turn the layers on and off we will need to output a JavaScript function within our HTML document that can be called each time a checkbox is clicked which will set the VISIBILITY style attribute appropriately. Our function will be named ‘OnOff’ and looks like this:

```
function OnOff(yr) {
         if(document.all['CB' + yr].checked==true) {
            document.all['G' + yr].style.visibility='visible';}
         else {document.all['G' + yr].style.visibility='hidden';}
}
```

This simple function is passed the year as a parameter (yr). It then uses an IF statement to check the current status of the checkbox being clicked and sets the VISIBILITY style attribute to ‘visible’ if the checkbox checked or ‘hidden’ if the checkbox is cleared. This will result in an on/off functionality for the graphics as the Checkboxes are checked and unchecked.

CREATING SOME TEST DATA

First we will need to create some sample data to test with. The following piece of code will create a simple data member containing an observation for each state. Each observation will have two numeric variables. VALUE will contain a static value of 1 and YEAR will have a randomly assigned year between 2001 and 2003. The code looks like this:

```
proc freq data=maps.states (keep=state);
     tables state / noprint out=freq1 (keep=state);
data testdata;
     retain value 1;
     set freq1;
     year = put(int(ranuni(0)*3)+1,1.)+2000;
run;
```

SAS/GRAPH, PROC REPORT AND ODS

Now let’s look at the two types of output we will be incorporating into our web documents and how they can be placed side-by-side on a web page.

GRAPHIC OUTPUT

We will need to create a series of U.S. plots, one for each year of data we will be displaying on our web page. We will also need to create a blank U.S. map containing only the state outlines. Later we will be combining, or more precisely overlaying, these plots using HTML constructs within our web document. The most basic example of the SAS/Graph logic to be used is the version which creates the blank U.S. map:

```
filename gifout "BlankUS.gif";
goptions reset=all transparency display vsize=350pt hsize=350pt cback=cxFFFFFE
         device=gif noborder gsfmode=replace gsfname=gifout;
pattern1 v=s r=1 c=white;
proc gmap data=maps.us map=maps.us all;
     choro state / missing nolegend coutline=black;
     id state;
run;
quit;
filename gifout clear;
```

The above code will create a GIF image output file with the name “BlankUS.gif”. GIF output is used since it allows the use of transparent backgrounds which are important for our application since it allows the underlying layers to show through. The TRANSPARENCY option on the GOPTIONS statement tells SAS/Graph we want our GIF output to make use of this feature. The DEVICE option specifies that we are creating GIF output. The CBACK option sets the background color, which will be transparent, slightly off from white so that we can have white areas within our plot without they themselves becoming transparent. The VSIZE and HSIZE options set a fixed size for the plot output. This is, once again, so that everything aligns properly. The GSFNAME option points to the FILEREF where the output is to be written and the GSFMODE option says to replace the output file if it already exists. The PATTERN statement says that we want the plot to be filled with a solid white color.

Now we can modify the same code and place it within a macro %DO loop to generate a series of plots for each of the three years of data to be presented in our document:

```
proc format;
     value color
           2001 = 'red'
           2002 = 'blue'
           2003 = 'green';
run;
%do i=2001 %to 2003;
    filename gifout "Year&i..gif";
    goptions reset=all transparency display vsize=350pt hsize=350pt cback=cxFFFFFE
             device=gif noborder gsfmode=replace gsfname=gifout;
    pattern1 v=s r=1 c=%sysfunc(putn(&i,color.));
    proc gmap data=testdata (where=(year=&i)) map=maps.us all;
         choro value / missing nolegend coutline=black;
         id state;
    run;
    quit;
    filename gifout clear;
%end;
```

Note the highlighted changes in the above code. They include a %DO loop to create a plot for each year, a dynamically generated GIF file name (Year2001.gif, Year2002.gif, etc.), a WHERE clause to subset the data to be plotted and a format applied via a %SYSFUNC macro function to set the color for each year's data. The colors actually assigned are controlled with the PROC FORMAT preceding the macro loop.

TABULAR OUTPUT WITH PROC REPORT

To create a list of years with checkboxes we will use ODS and the following PROC REPORT code:

```
ods listing close;
ods html file=htmlout (no_top_matter no_bottom_matter);
proc report data=testdata nowindows style={frame=void rules=none};
     columns year;
     define  year / group ' ';
     compute year;
             style = "style={prehtml=""<input id='CB"    || compress(year)   ||
                     "' type='checkbox' onclick='OnOff(" || compress(year)   ||
                     ");'></input>"" foreground="        || put(year,color.) ||
                     " font_weight=bold}";
            call define('year','STYLE',style);
     endcomp;
     title1 "<b><u>Data For Years 2001-2003</u></b>";
run;
ods html close;
ods listing;
```

The ODS HTML statement opens our HTML destination and points the output to the file referenced by the HTMLOUT fileref. The NO_TOP_MATTER and NO_BOTTOM_MATTER options tell ODS to only output the PROC REPORT results without adding the customary HTML page markup before and after it (we will be adding this ourselves using DATA steps). The STYLE option on the PROC REPORT statement says there are to be no borders around any of the report output. The COMPUTE block is what is doing most of the work here. The PREHTML style attribute says to include the specified text before the cell contents. Here is where we add the CHECKBOX markup to be inserted before the year value in our output. In HTML checkboxes are defined using the <INPUT> tag. This is done by adding the TYPE='CHECKBOX' attribute. Notice the ID= attribute constructed as part of this tag. It will result in an ID attribute in the form of, for example, "CB2001". These will match the CHECKBOX tag IDs referenced by our JavaScript function. The ONCLICK attribute of the <INPUT> tag says our JavaScript function 'OnOff' is to be executed whenever the checkbox is clicked and the value of YEAR is passed to it as a parameter. The FOREGROUND= style attribute and the PUT function set the text color of the output to match the color assigned in our earlier GMAP steps. The CALL DEFINE executes a style definition and assigns it to the YEAR column in our output. Finally we have added the HTML <B> and <U> tags to the TITLE definition to make the text bold and underlined, respectively.

COMBINING OUR PLOTS AND TABLE ON ONE PAGE

Although there several ways to accomplish this, including the creative use of style specifications, for the purpose of our example we are going to use the KISS method and use a DATA step to enclose our graphic and report output within an HTML table in a form similar to this:

```
<table>
   <tr>
      <td>
         ***<DIV> tags containing GMAP output***
      </td>
      <td>
         ***PROC REPORT output***
      </td>
   </tr>
</table>
```

ASSEMBLING THE PROGRAM

Here is a look at our final, assembled program:

```
/*** Create some test data. ***/
proc freq data=maps.states (keep=state);
     tables state / noprint out=freq1 (keep=state);
data testdata;
     retain value 1;
     set freq1;
     year = put(int(ranuni(0)*3)+1,1.)+2000;
run;
/*** Create blank U.S. plot. ***/
filename gifout "BlankUS.gif";
goptions reset=all transparency display vsize=350pt hsize=350pt cback=cxFFFFFE
         device=gif noborder gsfmode=replace gsfname=gifout;
pattern1 c=white v=s r=99;
proc gmap data=maps.us map=maps.us all;
     choro state / missing nolegend coutline=black;
     id state;
run;
quit;
filename gifout clear;
/*** Create plots for each year. ***/
proc format;
     value color 2001 = 'red'
                 2002 = 'blue'
                 2003 = 'green';
run;
%do i=2001 %to 2003;
    filename gifout "year&i..gif";
    goptions reset=all transparency display vsize=350pt hsize=350pt cback=cxFFFFFE
             device=gif noborder gsfmode=replace gsfname=gifout;
    pattern1 v=s r=1 c=%sysfunc(putn(&i,color.));
    proc gmap data=testdata (where=(year=&i)) map=maps.us all;
         choro value / missing nolegend coutline=black;
         id state;
    run;
    quit;
    filename gifout clear;
```

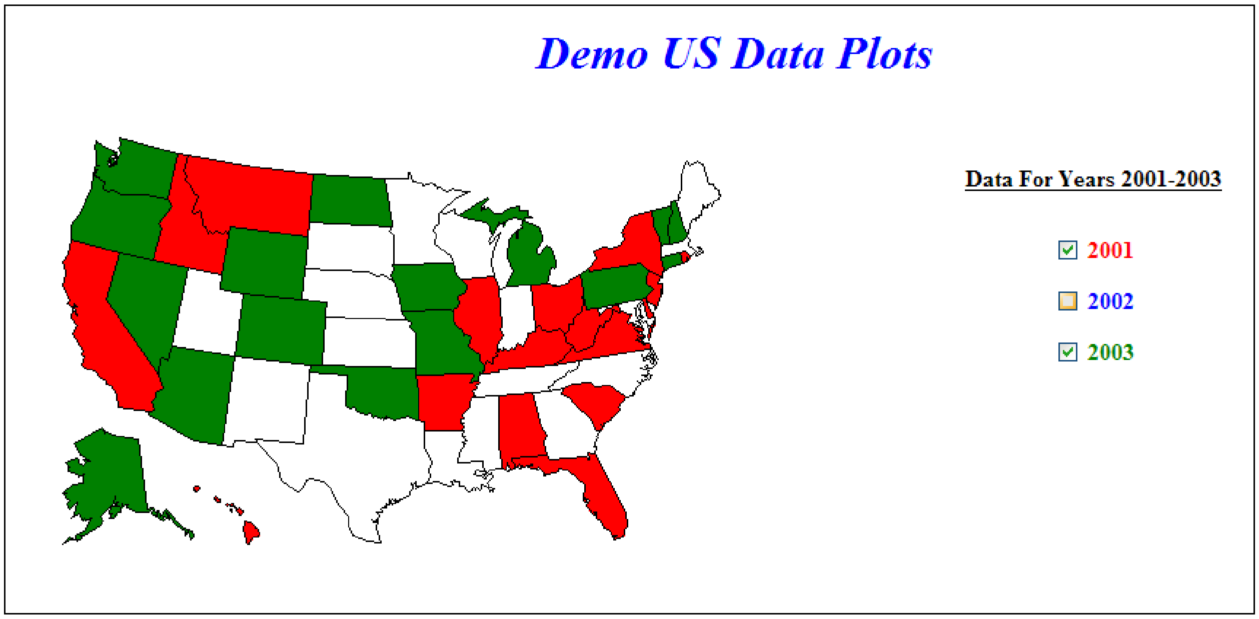
```
%end;
/*** Start the HTML document. ***/
data _null_;
     file "Demo.html";
     put;
run;
filename htmlout "Demo.html" mod;
data _null_;
     file htmlout;
     put "<html>" /
         "<head>" /
         "<script language='javascript'>" /
         "prevg=999" /
         "function OnOff(yr) {" /
         "         if(document.all['CB' + yr].checked==true) {" /
         "            document.all['G' + yr].style.visibility='visible';" /
         "            }" /
         "         else {" /
         "              document.all['G' + yr].style.visibility='hidden';" /
         "              }" /
         "}" /
         "</script>" /
         "<style>" /
         "TD {text-align: center;}" /
         "H1 {text-align: center; color: blue; font-style: italic;}" /
         "</style>" /
         "</head>" /
         "<body>" /
         "<h1>Demo US Data Plots</h1>" /
         "<table cellpadding=0 cellspacing=0 width='100%' height='100%' frame='void'
                 rules='none'>" /
         "<tr>" /
         "<td width='50%'>" /
         "<div id=""US"" align=""left"" valign=""top"" style=""position:absolute;
               top:0; left:35; z-index: 1;""><img src=""BlankUS.gif""></div>" /
         %do i=2001 %to 2003;
             "<div id=""G&i"" align=""left"" valign=""top""
             style=""position:absolute; top:0; left:35; z-index: 2;
                     visibility:hidden;""><img src=""year&i..gif""></div>" /
         %end;
         "</td>" /
         "<td width='50%' valign='top' align='center'>" /
         "<br><br>";
run;
/*** List years with Checkboxes using Proc Report. ***/
ods listing close;
ods html file=htmlout (no_top_matter no_bottom_matter);
proc report data=testdata nowindows style={frame=void rules=none};
     columns year;
     define  year / group ' ';
     compute year;
             style = "style={prehtml=""<input id='CB'  || compress(year)   || "'
                      type='checkbox' onclick='OnOff(" || compress(year)   ||
                     ");'></input>"" foreground="      || put(year,color.) || "
                      font_weight=bold}";
            call define('year','STYLE',style);
     endcomp;
     title1 "<b><u>Data For Years 2001-2003</u></b>";
run;
ods html close;
ods listing;
```

```
/*** Finish the HTML document. ***/
data _null_;
     file htmlout;
     put "</td>" /
         "</tr>" /
         "</table>" /
         "</body>" /
         "</html>";
run;
filename htmlout clear;
```

In the above code we have used a variety of techniques to tailor our output. A page title (the <H1> tag) has been added to the document by the DATA step. A style definition to control the title's appearance is also added within the HTML <STYLE> tags by the DATA step. The PROC REPORT which creates our list of checkboxes also contains a TITLE statement to provide a heading for the table which in turn has imbedded within it HTML tags (<B> and <U>) to make the heading bold and underlined. SAS/Graph creates the plots and macro logic is used to create multiple versions of the graphics. HTML tables place the output side by side and <DIV> tags are used to define layers whose visibility is controlled with a JavaScript function placed in our document by the DATA step and referenced in our web page by the checkboxes added to our output by the PROC REPORT's CALL DEFINE function.

THE FINAL OUTPUT

Here is a view of our final output:



By clicking on the desired year's checkbox the corresponding year is displayed or removed from the map

EXPANDING ON THIS EXAMPLE

Whereas this example is very simple and it's application extremely limited it can be seen that there is the potential for much more complicated applications to be built using these tools. In the above code much of the data and controls are hard coded. One obvious improvement would be the addition of logic to make the graph and checkbox creation data-dependant. More complicated layering schemes could be implemented as well as drill-down capabilities. Additionally, some of the HTML stylesheet and page layout controls could be done with ODS styles instead.

SUMMARY

There are multiple ways to create and customize HTML output with SAS. Much more sophisticated approaches could be used than are presented here. Using a combination of techniques virtually any type of web page functionality could be implemented in your SAS output. Even though the example shown in this paper has very limited application it can still be seen that the basic tools needed for creating interactive web documents are not particularly difficult to use and can be very flexible.

CONTACT INFORMATION

Your comments and questions are valued. Feel free to contact the author at:

E-mail: SAS@JamesRibble.com
Web: SAS.JamesRibble.com