Paper 1673-2014

Enhance the SAS® ODS HTML Output with JavaScript

Yu Fu, Oklahoma State Department of Health;
Chao Huang, Oklahoma State University;

ABSTRACT

For data analysts, one of the most important steps after manipulating and analyzing the dataset is to create a report for it. Nowadays, many statistics tables, reports are generated as HTML files and can be easily accessed through internet. However, the SAS® Output Delivery System (ODS) HTML output has many limitations on interacting with users. In this paper, we introduce a method to enhance the traditional ODS HTML output by using jQuery (a JavaScript library). A macro was developed to implement this idea. Compared to the standard HTML output, this macro can add sort, pagination, search and even drilldown function to the ODS HTML output file.

INTRODUCTION

jQuery is a fast, small, and feature-rich JavaScript library which provides an easy-to-use API to do things like DOM element selections, traversal, modification and so on. Used by over 65% of the 10,000 most visited websites, jQuery has become the most popular JavaScript library in use today. The paper will introduce a method to use DataTables(a plug-in for the jQuery) and embedded jQuery-based JavaScritpt in ODS HTML output to provide the data table in html output with sort, paging, search and drilldown function. All SAS code was wrapped within a macro program. In the following sections, we would explain this method step by step.

EMBED JAVASCRIPT INTO A ODS HTML OUTPUT

There are many ways you can use to inject your JavaScript into a HTML file. Some people like using FILE and PUT Statement to generate the HTML with JavaScript directly. This method is very flexible because you can add any HTML element you need into the file. However, it would require you have the knowledge of HTML language and also lose many good features that provided by ODS.

Therefore, in this paper we use the PREHTML= and POSTHTML= attributes within the Style element body to add JavaScript and HTML code to ODS output. PREHTML attribute adds code to the beginning of a file while POSTHTML attribute adds code to the end of the file. The example code is shown as below.

```
PROC TEMPLATE;
   DEFINE STYLE MYSTYLE;
       PARENT=STYLES.DEFAULT;
       STYLE BODY FROM BODY /
       PREHTML='<SCRIPT LANGUAGE=JAVASCRIPT
       TYPE="TEXT/JAVASCRIPT" SRC="http://code.jquery.com/jquery-1.9.1.js"></SCRIPT>';
   END;
RUN;


ODS HTML BODY='TEMP.HTML' STYLE=MYSTYLE;
   PROC PRINT DATA=SASHELP.CLASS;
   RUN;
ODS HTML CLOSE;
```

There is another way to add JavaScript to a file generated by ODS is to use the HEADTEXT= option for ODS Statement. However, values specified with the HEADTEXT= option that are limited to 256 characters. The exceed portion would be truncated. For PREHTML and POSTHTML, the same thing would not happen but SAS would give a warning message when the values are longer than 512 characters.

ADDING SORT, PAGINATION AND SEARCH TO HTML OUTPUT

The first part of enhancement is to add sort, pagination and search function to a traditional data table ODS HTML output. DataTables, a plug-in for the jQuery, was used to implement these functions. The following section of code embeds the DataTables into an ODS HTML output and configured it for the output data table.

```
%macro ods_html_prehtml(skey='idnum',dkey=idnum);
```

```
<style type='text/css'>
.dataTables_wrapper {
       POSITION: relative; ZOOM: 1; CLEAR: both
}
.dataTables_length {
       WIDTH: 40%; FLOAT: left;TEXT-ALIGN: LEFT;
}
.dataTables_filter {
       TEXT-ALIGN: right; WIDTH: 50%; FLOAT: right
}
.dataTables_info {
       WIDTH: 50%; FLOAT: left;TEXT-ALIGN: LEFT;
}
.dataTables_paginate {
       TEXT-ALIGN: right; FLOAT: right
}
TABLE.display {
       MARGIN: 0px auto; WIDTH: 100%; CLEAR: both
}
</style>
```

```
<script src='http://code.jquery.com/jquery-1.9.1.js'></script>
<script src='http://datatables.net/download/build/jquery.dataTables.nightly.js'>
</script>
```

```
<script>$(document).ready(

       function(){
              $('.table').attr('id','tablelist').addClass('display');
              $('.table').dataTable();

       }
);
</script>
```

```
%mend ods_html_prehtml;
```

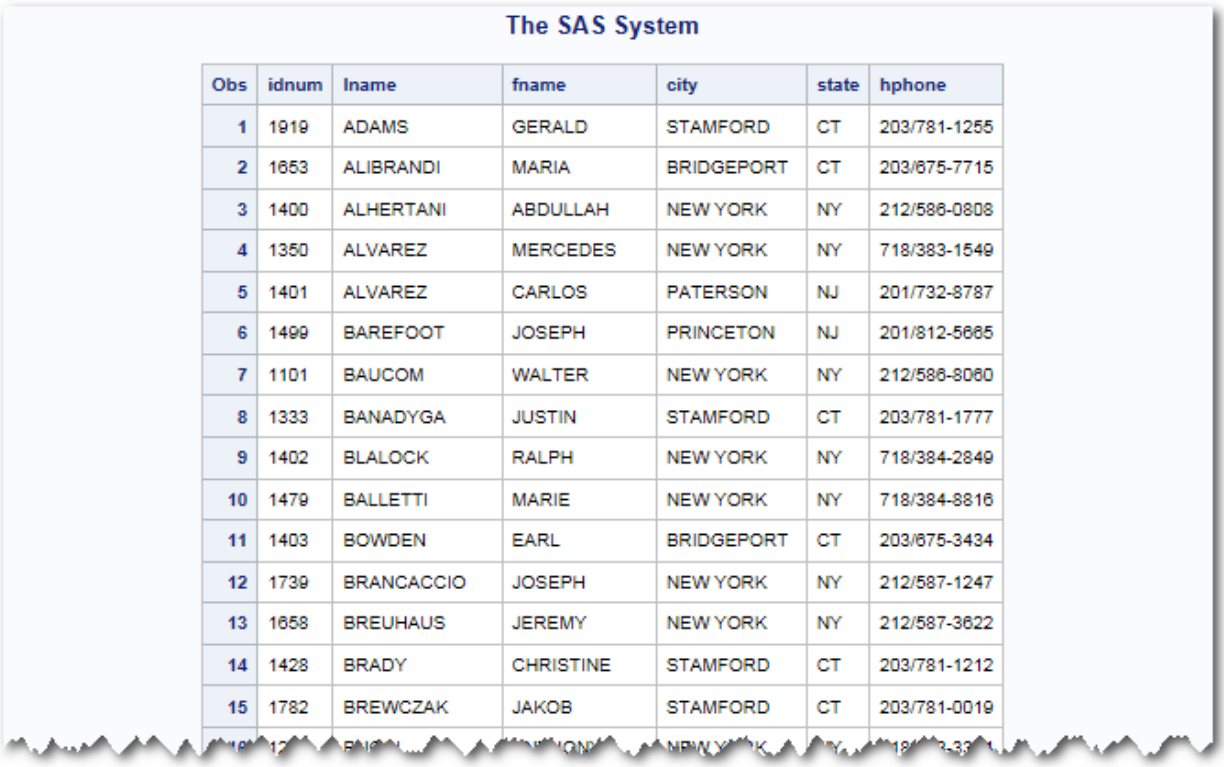
In this portion of code, we encapsulated all JavaScript and Cascading Style Sheets (CSS) into a macro. The return value of the macro would be assigned the PREHTML attribute.

The CSS code in the first box is used to set up the position for the new created HTML items like link, dropdown list, textbox and the like. Without these CSS, the new page would be in a mess. These CSS can be also put into a separated file and then referenced in the HTML file.

The code in the second box is used to include the jQuery and DataTables JavaScript into the new page. All these JavaScript will be downloaded from the internet when you open the ODS HTML output. That means you don't need to have these JavaScript on your local driver before you run the SAS code. What you need is to make sure the running computer is connected to the network. However, you can also add these JavaScript libs from a local path.

The JavaScript code in the third box will be run when the HTML file is fully loaded. These codes create some HTML TABLE attributes which are required by the DataTables. Then the DataTables is applied on the data table generated by the SAS ODS HTML.

The following two figures show the difference between before and after using the DataTables.



The SAS System

Obs	idnum	lname	fname	city	state	hphone
1	1919	ADAMS	GERALD	STAMFORD	CT	203/781-1255
2	1653	ALIBRANDI	MARIA	BRIDGEPORT	CT	203/675-7715
3	1400	ALHERTANI	ABDULLAH	NEW YORK	NY	212/586-0808
4	1350	ALVAREZ	MERCEDES	NEW YORK	NY	718/383-1549
5	1401	ALVAREZ	CARLOS	PATERSON	NJ	201/732-8787
6	1499	BAREFOOT	JOSEPH	PRINCETON	NJ	201/812-5665
7	1101	BAUCOM	WALTER	NEW YORK	NY	212/586-8060
8	1333	BANADYGA	JUSTIN	STAMFORD	CT	203/781-1777
9	1402	BLALOCK	RALPH	NEW YORK	NY	718/384-2849
10	1479	BALLETTI	MARIE	NEW YORK	NY	718/384-8816
11	1403	BOWDEN	EARL	BRIDGEPORT	CT	203/675-3434
12	1739	BRANCACCIO	JOSEPH	NEW YORK	NY	212/587-1247
13	1658	BREUHAUS	JEREMY	NEW YORK	NY	212/587-3622
14	1428	BRADY	CHRISTINE	STAMFORD	CT	203/781-1212
15	1782	BREWCZAK	JAKOB	STAMFORD	CT	203/781-0019

Figure 1 HTML Output before Using DataTables

The SAS System

Figure 2 HTML Output after Using DataTables Sorted by idnum

ADDING DRILLDOWN TO HTML OUTPUT

SAS has some build-in function to implement the drilldown for graph or html output. However, the detailed drilldown contents are generated with the main HTML output file and then referenced through HREF attribute. That means if the final HTML output has 1000 items which can be drilled down, the program have to generate 1000 drilldown HTML

pages. In this paper, we use another method to implement the same function without generating such many drilldown pages.

The first step to implement this function is to output the detailed data into a JavaScript Object Notation (JSON) file. JSON is a text-based open standard designed for human-readable data interchange. Derived from the JavaScript scripting language, JSON file is very easy to be manipulated by JavaScript. In the newest Base SAS(R) 9.4, a data set can be easily exported to a JSON file through JSON Procedure. In this paper, we still show you how to do it in Base SAS 9.3.

```
%macro genDetailJSON(detail=payroll,path=c:\temp);
ods output variables=DIC;
ods listing close;
proc contents data=&detail;
run;

proc sort data = dic;
      by num;
run;

data _NULL_ ;
      set dic end=last;
      call symputx("var"||left(_n_),variable,'L');
      if last then call symputx("num",_n_);
run;

DATA _NULL_;
    file "&path.\detail.JSON";
    set payroll end=lastrec;
    if _N_ eq 1 then do;
        put '[';
    end;
      put '{';
      %do i=1 %to &num;
            %if &i = &num %then %do;
            put '"' "&&var&i" '":"' &&var&i '"';
            %end;
            %else %do;
            put '"' "&&var&i" '":"' &&var&i '",';
            %end;
      %end;
      put '}';
    if lastrec eq 1 then do;
        put ']';
    end;
    else do;
        put ',';
    end;
RUN;
%mend genDetailJSON;
```

The macro above transfer a SAS data set into a local JSON file. FILE Statement specified the JSON file for the current out file for the PUT Statement.

The second step is to use JavaScript to fetch the required drill down detailed data from the JSON file. Most of the methods will redirect the user to another page to display the detailed data. Here, we use jQuery UI to display the detailed data within the summary page. Beside the code we introduced above for importing and using DataTables, the code below is added to PREHTML attribute to import the jQuery UI and fetch the detailed data from JSON file.

```
<script src='http://code.jquery.com/ui/1.10.3/jquery-ui.js'></script>
<link rel='stylesheet'
href='http://code.jquery.com/ui/1.10.3/themes/smoothness/jquery-ui.css' />
```

```
<script>$(document).ready(
      function(){
      var num;
      $('.table').children('thead').children('tr').first().children('th')
      .each(function(index) {if($(this).text()==&skey){num = index}});
      $('.table').children('tbody').children('tr').
      each(function(){$(this).children('td').eq(num-1).each(function(){
      var link = $('<a/>').attr('href','#').text($(this).text()).
      click(function(e){$('#dialog-modal').dialog('open');
      $.getJSON('detail.JSON',function(data){
            $('#dialog-modal').empty();
            var itemn = 0;
            $.each(data, function(i,item){
            key = $(e.target).text();
            if(item.&dkey..replace(/\s+/g, '')==key){
                  var hrow;
                  var drow;
                  if (itemn==0){
                  mytable = $('<table></table>').attr({ id: 'detailData' });
                  mytable.appendTo('#dialog-modal');
                  hrow = $('<tr></tr>').css('border','2px solid red')
                  .appendTo(mytable);
            }
            drow = $('<tr></tr>').appendTo(mytable);
            $.each(item, function(k, v) {
                  if(itemn==0){
                  var th = $('<th></th>').css('border','2px solid red')
                  .text(k).appendTo(hrow);
            }
            var tr = $('<td></td>').css('border','2px solid red')
            .text(v).appendTo(drow);
            });
            itemn = itemn + 1;}
                        });});
            });
            $(this).html(link);
      })}););
</script>
```

The code in the first box was used to import the jQuery UI JavaScript and CSS file from the internet. The rest of JavaScript codes create a link for each item which can drill down and also add response JavaScript functions for these links to fetch the detailed data from the JSON file.

In order to display the detailed data within the summary page, a hidden HTML DIV was created for jQuery UI to show and hidden a modal dialog in the page. The additional HTML code was included in the POSTHTML attribute. The portion of code is shown as below.

```
%macro ods_html_posthtml;
<div id='dialog-modal' title='Drill Down Table'>
</div>
%mend ods_html_posthtml;
```

```
PROC TEMPLATE;
   DEFINE STYLE mystyle;
         PARENT=STYLES.DEFAULT;
       STYLE BODY FROM BODY /
                  PREHTML="%ods_html_prehtml"
                  POSTHTML="%ods_html_posthtml";
   END;
RUN;
```

The final effect of the drill down function looks like the figure below.

Figure 3 Drill down Detailed Table

HOW TO USE THE MACRO

This section will teach you how to use this handy macro. The first step is to include the macro sas file in your sas file. The following code is used to bring the macro file into your current sas program.

```
%INCLUDE "datatable.SAS";
```

The path of the macro sas file should be modified based on where it locates. The parameters for this macro are listed as below.

path

is the directory where you want to output the final html file.
Note: when you are using the macro with drilldown function, the generated json file is also stored in this directory.

body

is the name of the generated html file.

summary

is the data set name for the summary data set.

detail

is the data set name for the detail data set.

Note: The data set name for *summary* or *detail* can be a one-level name if it is in the WORK lib (for example, PAYROLL), a two-level name (for example, IN.PAYROLL)

skey

is a variable name in summary data set.

dkey

is a variable name in detail data set.

Note: The macro uses *skey* and *dkey* to relate an observation in summary data set to observations in detail data set.

There are two ways to use this macro. One is to use without drill down function and the other is to use with drill down function. The two examples below are used to show you these two methods.

Example 1: Without drill down function

To use the macro without drill down, you don't need to assign any value to *detail*, *skey* and *dkey*.

```
%datatable(path=c:\temp,body=JQUERY_SORT.HTML,summary=sashelp.cars);
```

Example 2: With drill down function

The variable name assigned to *skey* is in a single quotation, the variable name assigned to *dkey* is not in a quotation.

```
proc sql;
      create table car_sum as select distinct(make),origin,count(*) as number
      from sashelp.cars group by make;
      create table car_detail as select
      Make,model,type,msrp,invoice,mpg_city,mpg_highway from sashelp.cars;
quit;

%datatable(path=c:\temp,body=JQUERY_SORT.HTML,summary=car_sum,detail=car_deta
il,skey='Make',dkey=Make);
```

LIMITATIONS

All these functions mentioned above are only tested under IE 8, IE10 and Firefox26. Using different or early version of web browser (Chrome etc.) may get JavaScript running error or find the final page not displayed properly. This compatible issue can be solved by modifying some portion of CSS or JavaScript codes.

CONCLUSION

In this paper, a new method is introduced to add sorting, paging, search and drill down functions onto a SAS ODS HTML output by using JavaScript. This method provides an enhancement to the current SAS ODS HTML system and let SAS users have more options to display their result in HTML output.

REFERENCES

- Style Attributes and Their Values [Internet]. [cited 2014 Jan 26]. Available at http://support.sas.com/documentation/cdl/en/odsug/61723/HTML/default/viewer.htm#a002972093.htm#a002978379
- How can I add a GIF file at the beginning of my ODS HTML output? [Internet]. [cited 2014 Jan 26]. Available at http://support.sas.com/kb/23/389.html
- Huang C. Make all SAS tables sortable in the output HTML [Internet] [cited 2014 Jan 26]. Available at http://www.sasanalysis.com/2013/01/make-all-sas-tables-sortable-in-output.html
- Andrew Z. Easy sortable HTML tables in SAS ODS [Internet]. [cited 2014 Jan 26]. Available at https://heuristically.wordpress.com/2013/01/17/sortable-html-tables-sas-ods/
- jQuery API document[Internet]. [cited 2014 Feb 2]. Available at http://api.jquery.com/

CONTACT INFORMATION

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

Name: Yu Fu
Enterprise: Oklahoma State Department of Health
City, State ZIP: Oklahoma City, OK 73117
E-mail: yu.fu39@gmail.com

Name: Chao Huang
Enterprise: Oklahoma State University
City, State ZIP: Stillwater, OK. 74075
Email: hchao8@gmail.com
Web: www.sasanalysis.com

APPENDIX

The two data sets “payroll” and “staff” used in this paper could be found on sas website.

PAYROLL: https://support.sas.com/documentation/cdl/en/proc/61895/HTML/default/viewer.htm#a000146820.htm

STAFF: https://support.sas.com/documentation/cdl/en/proc/61895/HTML/default/viewer.htm#a000146824.htm

DATATABLE.SAS

```
option spool;

%macro ods_html_prehtml(skey='idnum',dkey=idnum);
/*cascading style sheet(css) section*/
/*css for sort and pagination html page*/
<style type='text/css'>
.dataTables_wrapper {
      POSITION: relative; ZOOM: 1; CLEAR: both
}
.dataTables_length {
      WIDTH: 40%; FLOAT: left;TEXT-ALIGN: LEFT;
}
.dataTables_filter {
      TEXT-ALIGN: right; WIDTH: 50%; FLOAT: right
}
.dataTables_info {
      WIDTH: 50%; FLOAT: left;TEXT-ALIGN: LEFT;
}
.dataTables_paginate {
      TEXT-ALIGN: right; FLOAT: right
}
TABLE.display {
      MARGIN: 0px auto; WIDTH: 100%; CLEAR: both
}
</style>
/*css file for jQuery modal dialog which is used to contain drill down
table*/
<link rel='stylesheet'
href='http://code.jquery.com/ui/1.10.3/themes/smoothness/jquery-ui.css' />
/*cascading style sheet(css) section ends*/

/*javascript section*/
/*jQuery 1.9.1 javascript file*/
<script src='http://code.jquery.com/jquery-1.9.1.js'></script>
/*dataTable jQuery plug-in javascript file which is used for
sort,pagnination,search */
<script
src='http://datatables.net/download/build/jquery.dataTables.nightly.js'>
</script>
/*jQuery ui javascript file which is used for modal dialog of drill down
table*/
<script src='http://code.jquery.com/ui/1.10.3/jquery-ui.js'></script>

/*javascript functions for initializing summary table and drill down detail
table*/
<script>$(document).ready(
/*initialize drill down detail table*/
      function(){
```

```
      if(&skey.!='null'){
      var num;
/*find the index number for key column in summary table*/
      $('.table').children('thead').children('tr').first().children('th')
      .each(function(index) {if($(this).text()==&skey){num = index}});
/*create links for key column value and assign click functions for them*/
      $('.table').children('tbody').children('tr').each(function(){$(this)
      .children('td').eq(num-1).each(function(){
                  var link = $('<a/>').attr('href', '#')
                  .text($(this).text()).click(function(e){
                        $('#dialog-modal').dialog('open');
                        $.getJSON('detail.JSON',function(data){
                        $('#dialog-modal').empty();
                        var itemn = 0;
                        $.each(data, function(i,item){
                        key = $(e.target).text();
                        if(item.&dkey..replace(/\s+/g, '')==key){
                              var hrow;
                              var drow;
                              if (itemn==0){
                              mytable = $('<table></table>')
                              .attr({ id: 'detailData' });
                              mytable.appendTo('#dialog-modal');
                              hrow = $('<tr></tr>')
                        .css('border','2px solid red').appendTo(mytable);
                              }
                              drow = $('<tr></tr>').appendTo(mytable);
                              $.each(item, function(k, v) {
                                    if(itemn==0){
                                    var th = $('<th></th>')
                                    .css('border','2px solid red')
                                    .text(k).appendTo(hrow);
                                    }
                                    var tr = $('<td></td>')
                                    .css('border','2px solid red')
                                    .text(v).appendTo(drow);
                              });
                              itemn = itemn + 1;
                        }

                                          });});
                  });
                  $(this).html(link);
            })});}
/*initialize summary table with sort,pagination and search*/
            $('.table').attr('id','tablelist').addClass('display');
            $('.table').dataTable();
/*initialize modal diglog of drill down detail table*/
        $(function() {
          $('#dialog-modal').dialog({
            autoOpen: false,
             height: 140,
             width:450,
            modal: true
          });
        });
      }
```

```
);
</script>
/*javascript section ends*/
%mend ods_html_prehtml;
/*create a div html element as the place holder for modal dialog of drill
down detail table*/
%macro ods_html_posthtml;
<div id='dialog-modal' title='Drill Down Table'>
</div>
%mend ods_html_posthtml;
/*macro to export a dataset to a json file*/
%macro genDetailJSON(detail=payroll,path=c:\temp);
ods output variables=DIC;
ods listing close;
proc contents data=&detail;
run;

proc sort data = dic;
      by num;
run;

data _NULL_ ;
      set dic end=last;
      call symputx("var"||left(_n_),variable,'L');
      if last then call symputx("num",_n_);
run;

DATA _NULL_;
    file "&path.\detail.JSON";
    set &detail. end=lastrec;
    if _N_ eq 1 then do;
        put '[';
    end;
      put '{';
      %do i=1 %to &num;
            %if &i = &num %then %do;
            put '"' "&&var&i" '":"' &&var&i '"';
            %end;
            %else %do;
            put '"' "&&var&i" '":"' &&var&i '",';
            %end;
      %end;
      put '}';
    if lastrec eq 1 then do;
        put ']';
    end;
    else do;
        put ',';
    end;
RUN;
%mend genDetailJSON;

ods path(prepend) work.mystyle(update);
/*the main macro to enhance a html table with sort,pagination,search and
drill down */
```

```
%macro
datatable(path=c:\temp,body=JQUERY_SORT_TEST.HTML,summary=staff,detail=,skey=
'null',dkey=null);
%if &detail ~= %then %do;
      %genDetailJSON(detail=&detail,path=&path);
%end;
PROC TEMPLATE;
   DEFINE STYLE mystyle;
         PARENT=STYLES.DEFAULT;
        STYLE BODY FROM BODY /
                   PREHTML="%ods_html_prehtml(skey=&skey,dkey=&dkey)"
                   POSTHTML="%ods_html_posthtml";
   END;
RUN;


ODS HTML PATH= "&path" BODY="&body" STYLE=mystyle metatext='HTTP-EQUIV="X-UA-
Compatible" CONTENT="IE=edge"';
PROC PRINT DATA=&summary;
RUN;
ODS HTML CLOSE;
%mend datatable;
```