Paper 024-30

Spicing Up SAS/IntrNet® Applications

Jonah P. Turner, United States Bureau of the Census, Washington, D.C.

ABSTRACT

With the advent of SAS/IntrNet software, SAS programmers can extend their applications online to cross-platform, global end-users. The paper "A Pinch of SAS®, a Fraction of HTML, and a Touch of JavaScript Serve Up a Grand Recipe," published in the SUGI 28 proceedings, provided an introductory approach to developing programs that function via the Web. The next paper in this series, "A Recipe for Success: Migrating SAS/AF® Applications to the Web Using SAS®, HTML, and JavaScript," issued for SUGI 29, expounded on the subject by offering a workable method for modernizing legacy SAS windowing programs. Whereas these both outline basic strategies for implementing Web-based systems, this third installment goes a step further by demonstrating a variety of techniques suitable for spicing up such SAS/IntrNet applications.

Programmers can add more flavor to their SAS/IntrNet applications using the same ingredients by way of enriching procedure output and enhancing interface functionality. In particular, by adding vertical scrollbars to resulting tables, conditionally formatting output data, and embedding dynamic links that point to additional information or even call upon other programs, it is possible to design a more user-friendly and comprehensible Web application. Developers can also capitalize on certain programming routines to integrate both the front- and back-end processes of an Internet application into a single SAS program, thereby fostering a system that is easier to manage and remains up to date. Ultimately, these different techniques provide for an application design that caters to the usability and efficiency needs of all end-users and programmers alike.

INTRODUCTION

The majority of SAS/IntrNet applications are made up of two components: an HTML graphical user interface (GUI) and a back-end SAS program. The front-end GUI usually consists of text fields, radio buttons, selection boxes, and other form elements that are designed to capture user input. JavaScript may also be embedded into an HTML document to validate the information that a user enters, which in effect helps to reduce server-side processing. After an HTML form has been properly completed and submitted, the subsequent Internet request is handled by the Application Dispatcher. Both pieces of the Dispatcher, namely, the Application Broker and the Application Server, work in part to invoke a particular SAS program. The front-end form field entries, which are then defined as global macro variables to the background SAS program, can be used to control program execution, as well as manipulate the output that is ultimately returned and rendered in the user's Web browser.

Although the development of such SAS/IntrNet applications is, for the most part, straightforward, it can be rather difficult to design programs that are more functional for end-users. The various types and styles of reports that are generated with these Web-based applications must be readable and comprehensible in order for them to be effective. Having to scroll through many pages of data can pose a burden on those users who need to focus their attention on particular groups of records. In some circumstances, these Web reports are too crowded with information, making it hard for users to point out noteworthy content. For such cases, displaying only vital data in scrollable regions with links to detailed sections would better suffice. Users also often have trouble identifying certain records that may be of greater interest. By conditionally formatting these observations with different fonts, sizes, colors, and images, the resulting Web reports become less complicated to read through and understand.

In addition, programmers are faced with many challenges in designing SAS/IntrNet applications that are more manageable on their end. Maintaining an HTML form to be consistent with the data that is used by the back-end program is a demanding, yet critical task. If the values that populate the HTML selection boxes, text fields, and other form elements do not correspond to those of the background data, users may find themselves running queries that wastefully consume system resources and even produce erroneous results. Moreover, whenever individual changes are made to the front-end HTML interface, the back-end SAS program, or the underlying data, it is often the case that every one of these components has to be adjusted accordingly. Thus, by dynamically building both the GUI and procedure output commands within a single SAS program, it is possible to create an application that is easier to manage and always current.

The content discussed herein will focus on different strategies for resolving the various usability and maintenance issues common among SAS applications that function online. It should be noted that the described techniques are not required for developing workable SAS/IntrNet systems; however, with such enhancements, both end-users and programmers will find these Web-based applications to be more efficient and effective in operation.

BACKGROUND

To get a better understanding of how SAS – in particular, the SAS Output Delivery System (ODS) – HTML, and JavaScript are used together in the design and implementation of different types of SAS/IntrNet applications, it is recommended that the reader review the two previous pieces in this series of papers on the topic of SAS and Web development. The paper "A Pinch of SAS®, a Fraction of HTML, and a Touch of JavaScript Serve Up a Grand Recipe" published in the SUGI 28 proceedings, provides an introductory approach to creating programs that function via the Internet. Its follow-up, "A Recipe for Success: Migrating SAS/AF® Applications to the Web Using SAS®, HTML, and JavaScript," issued for SUGI 29, expounds on the subject by suggesting a workable method for modernizing SAS windowing programs while still maintaining legacy SCL code. These papers offer comparable techniques for building interactive applications that bring to bear the power of SAS directly to Web users.

IMPLEMENTATION

• *Problem*

The example to be discussed involves a made-up SAS/IntrNet application referred to as the "Employee Annual Timesheet Electronic Reporting System" or "EATERS" for short. The front-end GUI consists of a form with a selection box containing a list of the names of all current Company X employees. The user, who is typically a manager or a member of the Human Resources Department (HRD), can choose a name from the drop-down list to generate a given employee's yearly time schedule. This resulting Web report includes a breakdown of the time the selected employee has spent in the office, away at lunch, and on leave (i.e. sick, personal, vacation) for each day of the year, along with any relevant notes. Despite the usefulness of this application in operation, there are several problems with the current setup. On the whole, the generated Web timesheet reports are simply too congested with data, making it difficult for users to pinpoint and absorb noteworthy content, such as those observations where an employee has recorded working overtime. It has also been a painstaking task for the developers to continually keep the hard-coded HTML selection box values up to date, considering the number of employees that coincides with Company X's high turnover rate, in addition to managing the front- and back-end portions of this programming assignment. These issues and other related problems are more closely discussed within the detailed sections below.

The HTML code used to fashion the front-end GUI and the original background SAS program designed to generate the employee annual time schedule reports are illustrated here with corresponding screenshots of resultant output:

Employee Annual Timesheet Electronic Reporting System: Front-end (eatgui.html)

```
<HTML>
<HEAD>
  <TITLE>EATERS</TITLE>
  <SCRIPT>
    /* Ensure that the user has made a selection before submission */
    function chkForm() {
      if (document.employees.employee.selectedIndex == 0) {
        alert("Please make a selection.") ;
        return false ;
      } else return true ;
    }
  </SCRIPT>
</HEAD>
<BODY>
  <CENTER>
  <H1>Employee Annual Timesheet Electronic Reporting Sytem</H1>
  <HR><BR><BR><BR>
  <FORM NAME=employees METHOD=get ACTION=
  "/sasscripts/broker.exe" onSubmit="return chkForm();">
    <SELECT NAME=employee>
      <OPTION VALUE=none SELECTED>(Please make a selection)
      <OPTION VALUE="John Adams">Adams, John
      <OPTION VALUE="Chester Arthur">Arthur, Chester
      …
      <OPTION VALUE="George Washington">Washington, George
      <OPTION VALUE="Woodrow Wilson">Wilson, Woodrow
    </SELECT>
    <INPUT TYPE=submit VALUE=Submit>
    <INPUT TYPE=hidden NAME=_program VALUE=hrprg.eatall.sas>
    <INPUT TYPE=hidden NAME=_service VALUE=default>
    <INPUT TYPE=hidden NAME=_debug VALUE=0>
  </FORM>
  </CENTER>
</BODY>
</HTML>
```

→

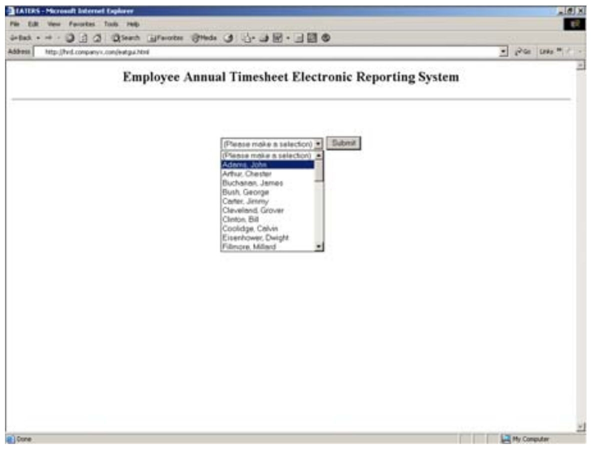
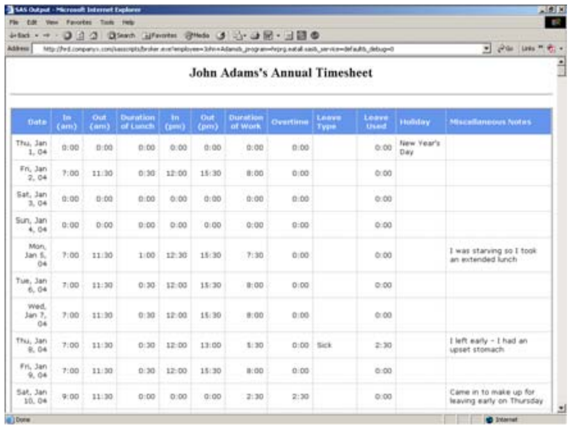

EATERS - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://hrd.companyx.com/eatgui.html
Go Links
Employee Annual Timesheet Electronic Reporting System
(Please make a selection)
Submit
(Please make a selection)
Adams, John
Arthur, Chester
Buchanan, James
Bush, George
Carter, Jimmy
Cleveland, Grover
Clinton, Bill
Coolidge, Calvin
Eisenhower, Dwight
Fillmore, Millard
Done
My Computer

Figure 1. Original GUI

<u>Employee Annual Timesheet Electronic Reporting System: Back-end (eatall.sas)</u>

```
%macro eatall ;

  /* Output the title */
  data _null_ ;
    file _webout ;
    put "<H2 ALIGN=center>&employee's
         Annual Timesheet</H2><HR>" ;
  run ;

  /* Output the annual timesheet for the user-selected employee */
  ods html body=_webout style=sasweb ;
  proc print data=hrdat.timesheets noobs label ;
    where employee="&employee" ;
    var date inam outam lunch inpm outpm work
         overtime leavetype leaveused holiday notes ;
    label date='Date'
           inam='In (am)'
           outam='Out (am)'
           lunch='Duration of Lunch'
           inpm='In (pm)'
           outpm='Out (pm)'
           work='Duration of Work'
           overtime='Overtime'
           leavetype='Leave Type'
           leaveused='Leave Used'
           holiday='Holiday'
           notes='Miscellaneous Notes' ;
    format date weekdate15. inam outam lunch inpm
             outpm work overtime leaveused hhmm. ;
  run ;
  ods html close ;

  /* Calculate the total work and overtime hours */
  proc means data=hrdat.timesheets ;
    where employee="&employee" ;
    var work overtime ;
    output out=totals1
            sum=totwork totovertime ;
  run ;
  data _null_ ;
    set totals1 ;
    call symput('totwork', totwork / 3600 ) ;
    call symput('totovertime', totovertime / 3600 ) ;
  run ;

  /* Calculate the total sick, personal, and vacation leave hours used */
  proc means data=hrdat.timesheets ;
    where employee="&employee" ;
    class leavetype ;
    var leaveused ;
    output out=totals2
            sum=leaveused ;
  run ;
  data _null_ ;
    set totals2 ;
    if leavetype='Sick' then call symput('totslu', leaveused / 3600 ) ;
    else if leavetype='Personal' then call symput('totplu', leaveused / 3600 ) ;
    else if leavetype='Vacation' then call symput('totvlu', leaveused / 3600 ) ;
  run ;

  /* Output the totals and a back button */
  data _null_ ;
    file _webout ;
    put '<CENTER>' ;
    put '<TABLE CELLSPACING=10 CELLPADING=10><TR>' ;
    put '<TD><B>Total Hours:</B></TD>' ;
    put "<TD><B>Billable: <I>&totwork</I></B></TD>" ;
    put "<TD><B>Overtime: <I>&totovertime</I></B></TD>" ;
    put "<TD><B>Sick Leave Used: <I>&totslu</I></B></TD>" ;
    put "<TD><B>Personal Leave Used: <I>&totplu</I></B></TD>" ;
    put "<TD><B>Vacation Leave Used: <I>&totvlu</I></B></TD>" ;
    put '</TR></TABLE>' ;
    put '<HR><INPUT TYPE=button VALUE=Back
          onClick="javascript:history.back();">' ;
    put '</CENTER>' ;
  run ;

%mend ;
%eatall ;
```

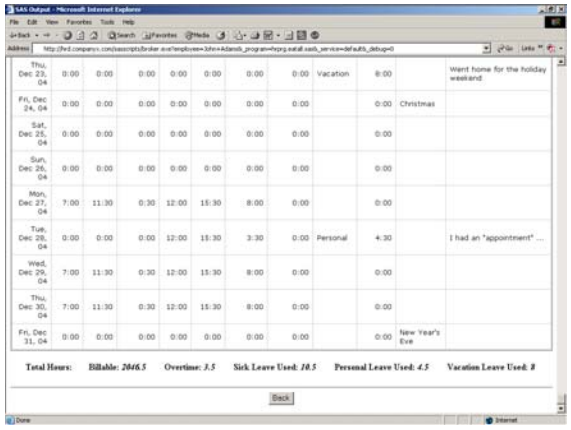
→


SAS Output - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://hrd.companyx.com/sasscripts/broker.exe?employee=John+Adams&_program=hrprg.eatall.sas&_service=default&_debug=0
Go Links
Done
Internet

John Adams's Annual Timesheet

Date	In (am)	Out (am)	Duration of Lunch	In (pm)	Out (pm)	Duration of Work	Overtime	Leave Type	Leave Used	Holiday	Miscellaneous Notes
Thu, Jan 1, 04	0:00	0:00	0:00	0:00	0:00	0:00	0:00		0:00	New Year's Day	
Fri, Jan 2, 04	7:00	11:30	0:30	12:00	15:30	8:00	0:00		0:00		
Sat, Jan 3, 04	0:00	0:00	0:00	0:00	0:00	0:00	0:00		0:00		
Sun, Jan 4, 04	0:00	0:00	0:00	0:00	0:00	0:00	0:00		0:00		
Mon, Jan 5, 04	7:00	11:30	1:00	12:30	15:30	7:30	0:00		0:00		I was starving so I took an extended lunch
Tue, Jan 6, 04	7:00	11:30	0:30	12:00	15:30	8:00	0:00		0:00		
Wed, Jan 7, 04	7:00	11:30	0:30	12:00	15:30	8:00	0:00		0:00		
Thu, Jan 8, 04	7:00	11:30	0:30	12:00	13:00	5:30	0:00	Sick	2:30		I left early - I had an upset stomach
Fri, Jan 9, 04	7:00	11:30	0:30	12:00	15:30	8:00	0:00		0:00		
Sat, Jan 10, 04	9:00	11:30	0:00	0:00	0:00	2:30	2:30		0:00		Came in to make up for leaving early on Thursday

Figure 2. Original output (top)


SAS Output - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://hrd.companyx.com/sasscripts/broker.exe?employee=John+Adams&_program=hrprg.eatall.sas&_service=default&_debug=0
Go Links
Back
Done
Internet

Thu, Dec 23, 04	0:00	0:00	0:00	0:00	0:00	0:00	0:00	Vacation	8:00		Went home for the holiday weekend
Fri, Dec 24, 04	0:00	0:00	0:00	0:00	0:00	0:00	0:00		0:00	Christmas	
Sat, Dec 25, 04	0:00	0:00	0:00	0:00	0:00	0:00	0:00		0:00		
Sun, Dec 26, 04	0:00	0:00	0:00	0:00	0:00	0:00	0:00		0:00		
Mon, Dec 27, 04	7:00	11:30	0:30	12:00	15:30	8:00	0:00		0:00		
Tue, Dec 28, 04	0:00	0:00	0:00	12:00	15:30	3:30	0:00	Personal	4:30		I had an "appointment" ...
Wed, Dec 29, 04	7:00	11:30	0:30	12:00	15:30	8:00	0:00		0:00		
Thu, Dec 30, 04	7:00	11:30	0:30	12:00	15:30	8:00	0:00		0:00		
Fri, Dec 31, 04	0:00	0:00	0:00	0:00	0:00	0:00	0:00		0:00	New Year's Eve	

Total Hours:	**Billable: *2046.5***	**Overtime: *3.5***	**Sick Leave Used: *10.5***	**Personal Leave Used: *4.5***	**Vacation Leave Used: *8***

Figure 3. Original output (bottom)

• *Solution*

By exploring the various programming issues separately and building the respective solutions into the application step by step, it will become clear as to how each technique can be exploited in real-world production systems that function online. The final version of this SAS/IntrNet application will work in a similar manner and produce comparable results; however, the enhancements made will provide for a system design that brings together the front- and back-end processes into a single SAS program – one that is easier to maintain and overall more functional.

▸ *Issue 0*

An easy, yet constructive technique for preparing a more flavorful application is to simply spice up its name. Surely, a title such as the "Employee Annual *Thyme*sheet Electronic Reporting System" would better suffice. ;)

▸ *Issue 1*

An apparent downside of the original application is that the time schedule Web reports, which include a record for every day of the year, take up too much space. Users who are primarily concerned with viewing the calculated totals find it cumbersome to have to scroll down to the bottom of the page each time to explore these results. Furthermore, depending on the length of the values assigned to certain variables (e.g. *date*, *holiday*, *notes*), the column widths seem to adjust accordingly; thus, a table may look different in the browser every day it gets updated.

When using the ODS, it can be a challenge to predict and control how procedure output will look when displayed in a Web browser given that SAS automatically generates the HTML code to be rendered. Developers can instead strategically place PUT statements within a DATA step to specify the exact HTML code that is to be written out to the source file. While some HTML code may be written out for each iteration of the DATA step, certain start tags and relevant attributes should only be applied during the first iteration and corresponding end tags on the final pass. For example, to output an entire data set, one would open and close an HTML *table* element on the first and last pass of a DATA step using the <TABLE> and </TABLE> tags, respectively, and affix a new row for each iteration:

```
data _null_ ;
  file file ;
  set dataset (end=last) ;
  if _n_=1 then put '<TABLE>' ;
  put '<TR><TD>' variable1 '</TD><TD>' variable2 '</TD><TD>' … '</TD><TD>' variableN  '</TD></TR>' ;
  if last then put '</TABLE>' ;
run ;
```

With some HTML knowledge and a little bit of creativity, developers can take advantage of this technique to completely control the output of their SAS/IntrNet programs. In this case, placing a time schedule report in a scrollable region would allow for the summary information listed below the table to be instantly observable. In conjunction with certain Cascading Style Sheet (CSS) properties, like *width*, *height*, and *overflow*, the <DIV> tag can be applied to define a scrollable region. To achieve precise column widths and specify other unique attributes to a column or group of columns, the <COLGROUP> and <COL> tags should be used as part of a *table* element. Here is a way for building an HTML table in a scrollable region that can be used in lieu of procedure output commands:

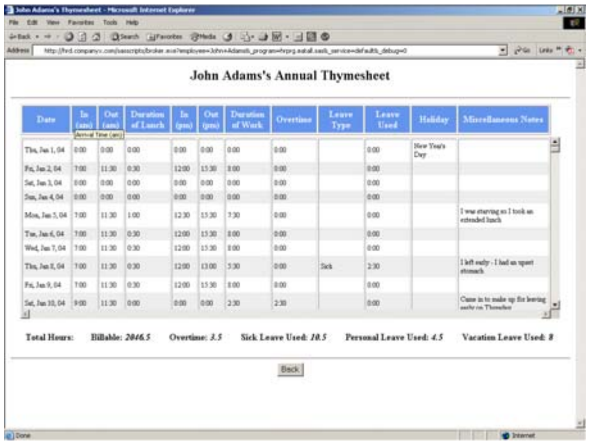
```
data _null_ ;
  file file ;
  set dataset (end=last) ;
  if _n_=1 then do ;
    put '<TABLE><TR><TD>' ;  /* This outer table ensures that both inner tables adjust to the browser window and are aligned */
    put '<TABLE FRAME=box>' ;  /* This inner table holds the column headers */
    put '<COLGROUP>' ;
    put '<COL SPAN=number_of_columns_to_span WIDTH=column_header_width>' ;
    /** Add another <COL> for each unique column grouping **/
    put '</COLGROUP>' ;
    put '<TR>' ;
    put '<TD>column_header_for_variable1</TD>' ;
    put '<TD>column_header_for_variable2</TD>' ;
    …
    put '<TD>column_header_for_variableN</TD>' ;
    put '</TR>' ;
    put '</TABLE>' ;
    put '</TD></TR><TR><TD>' ;
    put '<DIV STYLE="width:table_width;height:table_height;overflow:scroll;"><TABLE FRAME=box>' ;  /* This inner table holds the records */
    put '<COLGROUP>' ;
    put '<COL SPAN=number_of_columns_to_span WIDTH=column_width>' ;
    /** Add another <COL> for each unique column grouping **/
    put '</COLGROUP>' ;
  end ;
  put '<TR><TD>' variable1 '</TD><TD>' variable2 '</TD><TD>' … '</TD><TD>' variableN  '</TD></TR>' ;
  if last then do ;
    put '</TABLE></DIV>' ;
    put '</TD></TR></TABLE>' ;
  end ;
run ;
```

For the updated version of the EATERS, the PROC PRINT has been replaced with some DATA step processing to set the selected employee’s timesheet in a scrollable region with fixed dimensions. The user can now view all the summary information once the page loads without having to scroll to the bottom of the Web document. Longer records will also wrap inside respective table columns instead of forcing them to expand. With full control over the HTML code generated from this DATA step, it is possible to build other enhancements into the resulting table, such as mouseovers for column headers (using the *TITLE=* attribute) and alternating colors for subsequent rows of data:

Employee Annual Thymesheet Electronic Reporting System: Back-end (eatall.sas)

```
%macro eatall ;

  /* Output the employee’s annual timesheet in a scrollable region */
  data _null_ ;
    file _webout ;
    set hrdat.timesheets (end=last) ;
    if _n_=1 then do ;
      put '<HEAD>' ;
      put '<STYLE TYPE=text/css>' ;
      put '#Header {background-color:#6495ED;color:#FFFFFF;
          font-weight:bold;}' ;
      put '#RowA {background-color:#FFFFFF;font-size:12px;}' ;
      put '#RowB {background-color:#EEEEEE;font-size:12px;}' ;
      put '</STYLE>' ;
      put "<TITLE>&employee's Thymesheet</TITLE>" ;
      put '</HEAD>' ;
      put "<H2 ALIGN=center>&employee's
          Annual Thymesheet</H2><HR>" ;
      put '<TABLE ALIGN=center><TR><TD>' ;
      put '<TABLE CELLPADDING=4 FRAME=box>' ;
      put '<COLGROUP>' ;
      put '<COL SPAN=1 WIDTH=75>' ;
      put '<COL SPAN=2 WIDTH=35>' ;
      put '<COL SPAN=1 WIDTH=70>' ;
      put '<COL SPAN=2 WIDTH=35>' ;
      put '<COL SPAN=5 WIDTH=70>' ;
      put '<COL SPAN=1 WIDTH=150>' ;
      put '</COLGROUP>' ;
      put '<TR ID=Header ALIGN=center>' ;
      put '<TD TITLE="Date">Date</TD>' ;
      put '<TD TITLE="Arrival Time (am)">In<BR>(am)</TD>' ;
      put '<TD TITLE="Departure Time (am)">Out<BR>(am)</TD>' ;
      put '<TD TITLE="Total Lunch">Duration<BR>of Lunch</TD>' ;
      put '<TD TITLE="Arrival Time (pm)">In<BR>(pm)</TD>' ;
      put '<TD TITLE="Departure Time (pm)">Out<BR>(pm)</TD>' ;
      put '<TD TITLE="Total Work">Duration<BR>of Work</TD>' ;
      put '<TD TITLE="Total Overtime">Overtime</TD>' ;
      put '<TD TITLE="Type of Leave Used">Leave<BR>Type</TD>' ;
      put '<TD TITLE="Total Leave Used">Leave<BR>Used</TD>' ;
      put '<TD TITLE="National Holiday">Holiday</TD>' ;
      put '<TD TITLE="Extra Message">Miscellaneous Notes</TD>' ;
      put '</TR>' ;
      put '</TABLE>' ;
      put '</TD></TR><TR><TD>' ;
      put '<DIV STYLE="width:100%;height:320;overflow:scroll;">' ;
      put '<TABLE CELLPADDING=4 RULES=cols FRAME=box>' ;
      put '<COLGROUP>' ;
      put '<COL SPAN=1 WIDTH=75>' ;
      put '<COL SPAN=2 WIDTH=35>' ;
      put '<COL SPAN=1 WIDTH=70>' ;
      put '<COL SPAN=2 WIDTH=35>' ;
      put '<COL SPAN=5 WIDTH=70>' ;
      put '<COL SPAN=1 WIDTH=150>' ;
      put '</COLGROUP>' ;
    end ;
    if mod(_n_,2)=1 then put '<TR ID=RowA>' ;
    else put '<TR ID=RowB>' ;
    put '<TD>' date '</TD><TD>' inam '</TD><TD>' outam
        '</TD><TD>' lunch '</TD><TD>' inpm '</TD><TD>'
        outpm '</TD><TD>' work '</TD><TD>' overtime
        '</TD><TD>' leavetype '</TD><TD>' leaveused
        '</TD><TD>' holiday '</TD><TD>' notes '</TD></TR>' ;
    if last then do ;
      put '</TABLE></DIV>' ;
      put '</TD></TR></TABLE>' ;
    end ;
    format date weekdate15. inam outam lunch inpm
           outpm work overtime leaveused hhmm. ;
  run ;

  /** Calculate and output the totals in the same manner as before **/

%mend ;
%eatall ;
```

→



John Adams's Annual Thymesheet

Date	In (am)	Out (am)	Duration of Lunch	In (pm)	Out (pm)	Duration of Work	Overtime	Leave Type	Leave Used	Holiday	Miscellaneous Notes
Thu, Jan 1, 04	0:00	0:00	0:00	0:00	0:00	0:00	0:00		0:00	New Year's Day	
Fri, Jan 2, 04	7:00	11:30	0:30	12:00	15:30	8:00	0:00		0:00		
Sat, Jan 3, 04	0:00	0:00	0:00	0:00	0:00	0:00	0:00		0:00		
Sun, Jan 4, 04	0:00	0:00	0:00	0:00	0:00	0:00	0:00		0:00		
Mon, Jan 5, 04	7:00	11:30	1:00	12:30	15:30	7:30	0:00		0:00		I was starving so I took an extended lunch
Tue, Jan 6, 04	7:00	11:30	0:30	12:00	15:30	8:00	0:00		0:00		
Wed, Jan 7, 04	7:00	11:30	0:30	12:00	15:30	8:00	0:00		0:00		
Thu, Jan 8, 04	7:00	11:30	0:30	12:00	13:00	5:30	0:00	Sick	2:30		I left early - I had an upset stomach
Fri, Jan 9, 04	7:00	11:30	0:30	12:00	15:30	8:00	0:00		0:00		
Sat, Jan 10, 04	9:00	11:30	0:00	0:00	0:00	2:30	2:30		0:00		Came in to make up for leaving early on Thursday

Total Hours: **Billable:** ***2046.5*** **Overtime:** ***3.5*** **Sick Leave Used:** ***10.5*** **Personal Leave Used:** ***4.5*** **Vacation Leave Used:** ***8***

John Adams's Thymesheet - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://hrd.companyx.com/sasscripts/broker.exe?employee=John+Adams&_program=hrprg.eatall.sas&_service=default&_debug=0
Arrival Time (am)
Back
Done
Internet

Figure 4. Modified output with a scrollable region using the HTML *colgroup* and *col* elements

▸ *Issue 2*

Even though the EATERS was originally designed to cover all aspects of Company X employees' work schedules, it is not so easy for users to isolate information relevant to their individual needs. Generally speaking, there is too much information being displayed, which can be distracting for managers and HRD personnel. Whereas supervisors are mostly interested in knowing when an employee has taken an extended lunch break or worked less than a full 8-hour day, those from HRD typically use this tool to account for the overtime hours each staff member has earned.

Although it is quite easy to output a data set to the Web, generating a meaningful report can be a difficult task, especially if there is a great deal of information that needs to be displayed. An effective method for highlighting more important fields is to conditionally format output data so that users can easily and quickly discern such content. In other words, programmers can redefine particular records using different HTML tags and attributes that alter the color, size, and style of these values, rather than simply writing out observations in their original state, in order to put greater emphasis on certain output data. This can be done by creating temporary variables that not only maintain the values of the original variables, but also include embedded HTML to control how they should appear when rendered in a Web browser. For instance, a SAS/IntrNet application used to report all Company X employees' salaries would be more insightful if those greater than $100,000 were displayed in green, incomes less than $35,000 were italicized, and salaries equal to the median income of $55,000 were shown in larger text and bolded:

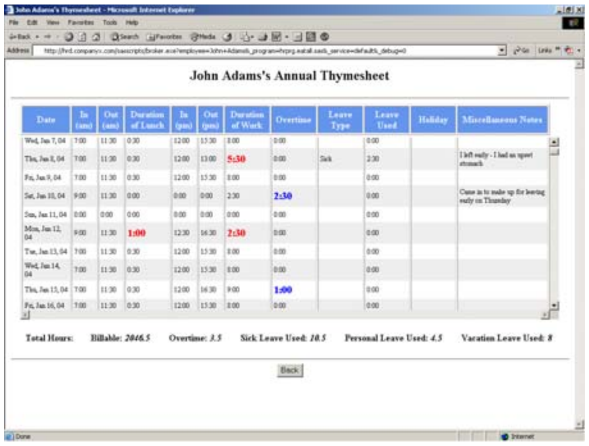
```
data salaries (drop=income rename=(tmpincome=income)) ;
   set hrdat.salaries ;
   /* Alter the color, size, and style of the output text for income based on its original value */
   if income > 100000 then tmpincome='<FONT COLOR=green>'||income||'</FONT>' ;
   else if income < 35000 then tmpincome='<I>'||income||'</I>' ;
   else if income = 55000 then tmpincome='<BIG><B>'||income||'</B></BIG>' ;
run ;
ods html body=_webout ;
proc print data=salaries ;  run ;
ods html close ;
```

For the EATERS, this technique could certainly be applied to satisfy the specific needs of users. Managers would be able to more quickly identify the days when an employee has taken an extended lunch break or worked less than a full 8-hour day if these records were highlighted in red, bold, and italicized text. Additionally, staff members from HRD would have an easier time sorting through the time schedule reports if records where overtime hours have been documented were displayed with blue, bold, and italicized text. In effect, all users should be able to isolate the vital information they need without much difficulty since these records will now appear in a more discernible fashion:

<u>Employee Annual Thymesheet Electronic Reporting System: Back-end (eatall.sas)</u>

```
%macro eatall ;
  /* Conditionally format records to be red/blue, bold, italicized text */
  data _null_ ;
    file _webout ;
    set hrdat.timesheets (end=last) ;
    if lunch > 1800 then tmplunch='<FONT COLOR=red>
      <B><I>'||put(lunch,hhmm.)||'</I></B></FONT>' ;
    else tmplunch=put(lunch,hhmm.) ;
    if (mod(date,7) ne 1 and mod(date,7) ne 2) then do ;
      if work < 28800 then tmpwork='<FONT COLOR=red>
        <B><I>'||put(work,hhmm.)||'</I></B></FONT>' ;
      else tmpwork=put(work,hhmm.) ;
    end ;
    else tmpwork=put(work,hhmm.) ;
    if overtime > 0 then tmpovertime='<FONT COLOR=blue>
      <B><I>'||put(overtime,hhmm.)||'</I></B></FONT>' ;
    else tmpovertime=put(overtime,hhmm.) ;
    if _n_=1 then do ;
        /** Build the headers and open the tables as before **/
    end ;
    if mod(_n_,2)=1 then put '<TR ID=RowA>' ;
    else put '<TR ID=RowB>' ;
    put '<TD>' date '</TD><TD>' inam '</TD><TD>' outam
        '</TD><TD>' tmplunch '</TD><TD>' inpm '</TD><TD>'
        outpm '</TD><TD>' tmpwork '</TD><TD>' tmpovertime
        '</TD><TD>' leavetype '</TD><TD>' leaveused
        '</TD><TD>' holiday '</TD><TD>' notes '</TD></TR>' ;
    if last then do ;
        /** Close the tables and build the footers as before **/
    end ;
    format date weekdate15. inam outam inpm outpm leaveused hhmm. ;
  run ;

  /** Calculate and output the totals in the same manner as before **/

%mend ;
%eatall ;
```

→



John Adams's Thymesheet - Microsoft Internet Explorer

http://hrd.companyx.com/saxscripts/broker.exe?employee=John+Adams&_program=hrprg.eatall.sas&_service=default&_debug=0

John Adams's Annual Thymesheet

Date	In (am)	Out (am)	Duration of Lunch	In (pm)	Out (pm)	Duration of Work	Overtime	Leave Type	Leave Used	Holiday	Miscellaneous Notes
Wed, Jan 7, 04	7:00	11:30	0:30	12:00	15:30	8:00	0:00		0:00		
Thu, Jan 8, 04	7:00	11:30	0:30	12:00	13:00	***5:30***	0:00	Sick	2:30		I left early - I had an upset stomach
Fri, Jan 9, 04	7:00	11:30	0:30	12:00	15:30	8:00	0:00		0:00		
Sat, Jan 10, 04	9:00	11:30	0:00	0:00	0:00	2:30	***2:30***		0:00		Came in to make up for leaving early on Thursday
Sun, Jan 11, 04	0:00	0:00	0:00	0:00	0:00	0:00	0:00		0:00		
Mon, Jan 12, 04	9:00	11:30	***1:00***	12:30	16:30	***2:30***	0:00		0:00		
Tue, Jan 13, 04	7:00	11:30	0:30	12:00	15:30	8:00	0:00		0:00		
Wed, Jan 14, 04	7:00	11:30	0:30	12:00	15:30	8:00	0:00		0:00		
Thu, Jan 15, 04	7:00	11:30	0:30	12:00	16:30	9:00	***1:00***		0:00		
Fri, Jan 16, 04	7:00	11:30	0:30	12:00	15:30	8:00	0:00		0:00		

Total Hours: **Billable:** ***2046.5*** **Overtime:** ***3.5*** **Sick Leave Used:** ***10.5*** **Personal Leave Used:** ***4.5*** **Vacation Leave Used:** ***8***

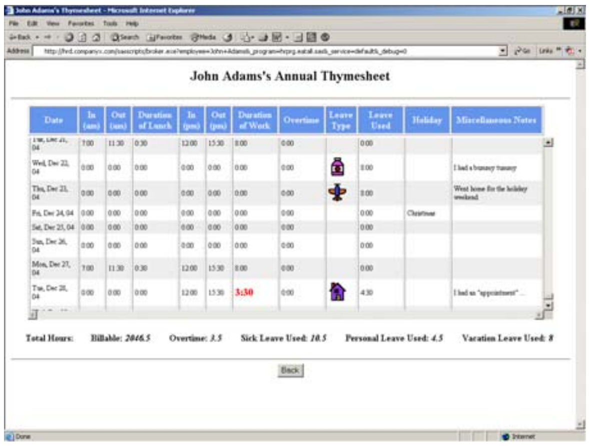
Figure 5. Modified output with conditionally formatted data using temporary variables where extended lunch breaks and short work days are denoted in red, bold, italicized text and recorded overtime hours are denoted in blue, bold, and italicized text

More can be accomplished with this technique of redefining variables to include embedded HTML than simply modifying the color, size, and style of text as it appears in a Web browser. In fact, programmers can mark up output data in any other manner that they are able to do when designing HTML interfaces. For instance, output text can be (conditionally) replaced with meaningful images to reduce space, offer clearer messages, or simply jazz up a Web report. Colorful icons to show the type of leave used help to make the time schedule reports more readable and fun:

Employee Annual Thymesheet Electronic Reporting System: Back-end (eatall.sas)

```
%macro eatall ;
  /* Conditionally format leavetype with images */
  data _null_ ;
    file _webout ;
    set hrdat.timesheets (end=last) ;
    if leavetype='Sick' then tmpleavetype=
      '<IMG SRC=/Images/sick.gif ALT=Sick>' ;
    else if leavetype='Personal' then tmpleavetype=
      '<IMG SRC=/Images/personal.gif ALT=Personal>' ;
    else if leavetype='Vacation' then tmpleavetype=
      '<IMG SRC=/Images/vacation.gif ALT=Vacation>' ;
    /** Conditionally format the other fields as before **/
    if _n_=1 then do ;
         /** Build the headers and open the tables as before **/
    end ;
    if mod(_n_,2)=1 then put '<TR ID=RowA>' ;
    else put '<TR ID=RowB>' ;
    put '<TD>' date '</TD><TD>' inam '</TD><TD>' outam
        '</TD><TD>' tmplunch '</TD><TD>' inpm '</TD><TD>'
        outpm '</TD><TD>' tmpwork '</TD><TD>' tmpovertime
        '</TD><TD>' tmpleavetype '</TD><TD>' leaveused
        '</TD><TD>' holiday '</TD><TD>' notes '</TD></TR>' ;
    if last then do ;
        /** Close the tables and build the footers as before **/
    end ;
    format date weekdate15. inam outam inpm outpm leaveused hhmm. ;
  run ;

  /** Calculate and output the totals in the same manner as before **/

%mend ;
%eatall ;
```

→



John Adams's Annual Thymesheet

Date	In (am)	Out (am)	Duration of Lunch	In (pm)	Out (pm)	Duration of Work	Overtime	Leave Type	Leave Used	Holiday	Miscellaneous Notes
Tue, Dec 21, 04	7:00	11:30	0:30	12:00	15:30	8:00	0:00		0:00		
Wed, Dec 22, 04	0:00	0:00	0:00	0:00	0:00	0:00	0:00	🧴	8:00		I had a bummy tummy
Thu, Dec 23, 04	0:00	0:00	0:00	0:00	0:00	0:00	0:00	✈	8:00		Went home for the holiday weekend
Fri, Dec 24, 04	0:00	0:00	0:00	0:00	0:00	0:00	0:00		0:00	Christmas	
Sat, Dec 25, 04	0:00	0:00	0:00	0:00	0:00	0:00	0:00		0:00		
Sun, Dec 26, 04	0:00	0:00	0:00	0:00	0:00	0:00	0:00		0:00		
Mon, Dec 27, 04	7:00	11:30	0:30	12:00	15:30	8:00	0:00		0:00		
Tue, Dec 28, 04	0:00	0:00	0:00	12:00	15:30	3:30	0:00	🏠	4:30		I had an "appointment" ...

Total Hours: Billable: *2046.5* Overtime: *3.5* Sick Leave Used: *10.5* Personal Leave Used: *4.5* Vacation Leave Used: *8*

Figure 6. Modified output with conditionally formatted data using temporary variables where "Leave Type" is denoted by these icons:

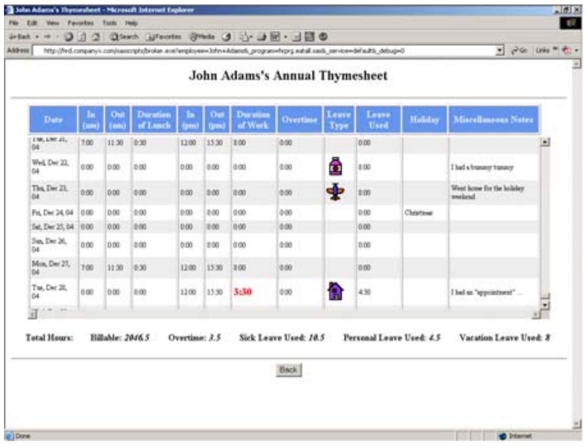
🧴 = sick 🏠 = personal ✈ = vacation

It is apparent that recreating variables with embedded HTML is an effective way to control how output data is to be displayed on the Web. An alternative technique that programmers can use to determine how data set records should be written out to an Internet source file is to build a SAS format consisting of HTML. The FORMAT procedure can instead be applied to replace certain values in the "Leave Type" column with the proper syntax for an HTML image:

Employee Annual Thymesheet Electronic Reporting System: Back-end (eatall.sas)

```
%macro eatall ;
  /* $leave is used to conditionally format leavetype with images */
  proc format ;
    value $leave
      'Sick'='<IMG SRC=/Images/sick.gif ALT=Sick>'
      'Personal'='<IMG SRC=/Images/personal.gif ALT=Personal>'
      'Vacation'='<IMG SRC=/Images/vacation.gif ALT=Vacation>' ;
  run ;

  data _null_ ;
    file _webout ;
    set hrdat.timesheets (end=last) ;
    /** Conditionally format the other fields as before **/
    if _n_=1 then do ;
        /** Build the headers and open the tables as before **/
    end ;
    if mod(_n_,2)=1 then put '<TR ID=RowA>' ;
    else put '<TR ID=RowB>' ;
    put '<TD>' date '</TD><TD>' inam '</TD><TD>' outam
        '</TD><TD>' tmplunch '</TD><TD>' inpm '</TD><TD>'
        outpm '</TD><TD>' tmpwork '</TD><TD>' tmpovertime
        '</TD><TD>' leavetype '</TD><TD>' leaveused
        '</TD><TD>' holiday '</TD><TD>' notes '</TD></TR>' ;
    if last then do ;
        /** Close the tables and build the footers as before **/
    end ;
    format date weekdate15. inam outam inpm outpm leaveused hhmm.
            leavetype $leave. ;
  run ;

  /** Calculate and output the totals in the same manner as before **/

%mend ;
%eatall ;
```

→



John Adams's Annual Thymesheet

Date	In (am)	Out (am)	Duration of Lunch	In (pm)	Out (pm)	Duration of Work	Overtime	Leave Type	Leave Used	Holiday	Miscellaneous Notes
Tue, Dec 21, 04	7:00	11:30	0:30	12:00	15:30	8:00	0:00		0:00		
Wed, Dec 22, 04	0:00	0:00	0:00	0:00	0:00	0:00	0:00	🧴	8:00		I had a bummy tummy
Thu, Dec 23, 04	0:00	0:00	0:00	0:00	0:00	0:00	0:00	✈	8:00		Went home for the holiday weekend
Fri, Dec 24, 04	0:00	0:00	0:00	0:00	0:00	0:00	0:00		0:00	Christmas	
Sat, Dec 25, 04	0:00	0:00	0:00	0:00	0:00	0:00	0:00		0:00		
Sun, Dec 26, 04	0:00	0:00	0:00	0:00	0:00	0:00	0:00		0:00		
Mon, Dec 27, 04	7:00	11:30	0:30	12:00	15:30	8:00	0:00		0:00		
Tue, Dec 28, 04	0:00	0:00	0:00	12:00	15:30	3:30	0:00	🏠	4:30		I had an "appointment" ...

Total Hours: Billable: *2046.5* Overtime: *3.5* Sick Leave Used: *10.5* Personal Leave Used: *4.5* Vacation Leave Used: *8*

Figure 7. Modified output with conditionally formatted data using PROC FORMAT where "Leave Type" is denoted by these icons:

🧴 = sick 🏠 = personal ✈ = vacation

These results are equivalent to those shown in Figure 6

▸ *Issue 3*

The above enhancements have helped to make the time schedule Web reports a little more understandable. Still, some issues with wasted space and overcrowded content exist on account of two rarely used fields. One column, which displays recurring national holidays, is almost always blank, and another field that includes miscellaneous notes left by the employee (e.g. “I left early – I had an upset stomach”) is usually empty; however, removing these fields altogether would not satisfy the requirements as most users need to view this information from time to time.

It is common for data that is illustrated in one report to be connected to data displayed in another. When users are able to view all this related content together, they have an easier time analyzing such information. Be that as it may, the more information that gets included in a report, the more crowded and less meaningful it becomes. To resolve this problem in the Web environment, developers can strategically include hyperlinks that point to supplementary information using the HTML *anchor* element. By redefining SAS variables with embedded HTML using the same technique discussed earlier, it is possible to render plain output data as links that point to additional content or even call upon other programs. For example, a SAS/IntrNet application that is used to generate a report listing every Company X employee may be more useful if the text for each person was actually a link to a personal profile page:

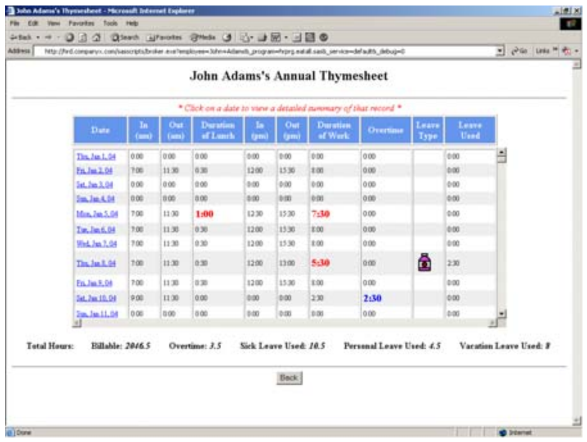
```
data staff (drop=person rename=(tmpperson=person)) ;
  set hrdat.staff ;
  /* Render each employee's name as a link to a personal profile document; e.g. Jonah Turner's name points to /HRD/profiles/JonahTurner.html */
  tmpperson='<A HREF="/HRD/profiles/'||compress(person)||'.html">'||person||'</A>' ;
run ;
ods html body=_webout ;
proc print data=staff ;  run ;
ods html close ;
```

In this case, removing two rarely populated fields would certainly help to reduce the amount of space that is wasted and, in turn, make the Web reports generated by the EATERS more readable. Since the users may still want to view this information on occasion, perhaps to remind themselves of an upcoming holiday or read what “excuse” an employee has marked down for leaving work early, it would be useful to have hyperlinks, one for each day, inserted in the table that point to the supplementary content. Rather than creating another field to contain such links, which would also lead to a loss of valuable space, programmers can instead strategically redefine existing variables with embedded HTML to construct them. For this example, the values already being displayed for the date field can be dynamically transformed into unique links that call upon another SAS/IntrNet program to display the extra information. That program will use the distinct employee and date parameters built into each hyperlink to perform a lookup of the record and return to the user’s browser a detailed summary report for the specific day that was chosen. With these hyperlinks in place, the original time schedule reports become less crowded as the fields that were regularly empty are now gone; yet, users can still view this extra content by clicking on the link for a particular day:

Employee Annual Thymesheet Electronic Reporting System: Back-end (eatall.sas)

```
%macro eatall ;

  /* Format date with hyperlinks that call upon another program */
  data _null_ ;
    file _webout ;
    set hrdat.timesheets (end=last) ;
    tmpdate='<A HREF="broker.exe?employee='||"&employee"||
    '&date='||trim(left(date))||'&_program=hrprg.eatone.sas
    &_service=default&_debug=0">'||put(date,weekdate15.)||'</A>' ;
    /** Conditionally format the other fields as before **/
    if _n_=1 then do ;
        /** Build the headers as before **/
        put '<FONT COLOR=red><I>* Click on a date to view
             a detailed summary of that record *</I></FONT>' ;
        /** Open the tables as before **/
    end ;
    if mod(_n_,2)=1 then put '<TR ID=RowA>' ;
    else put '<TR ID=RowB>' ;
    put '<TD>' tmpdate '</TD><TD>' inam '</TD><TD>' outam
        '</TD><TD>' tmplunch '</TD><TD>' inpm '</TD><TD>'
        outpm '</TD><TD>' tmpwork '</TD><TD>' tmpovertime
        '</TD><TD>' tmpleavetype '</TD><TD>' leaveused
        '</TD><TD>' holiday '</TD><TD>' notes '</TD></TR>' ;
    if last then do ;
        /** Close the tables and build the footers as before **/
    end ;
    format inam outam inpm outpm leaveused hhmm. ;
  run ;

  /** Calculate and output the totals in the same manner as before **/

%mend ;
%eatall ;
```

→


John Adams's Thymesheet - Microsoft Internet Explorer

Address http://hrd.companyx.com/sasscripts/broker.exe?employee=John+Adams&_program=hrprg.eatall.sas&_service=default&_debug=0

John Adams's Annual Thymesheet

* Click on a date to view a detailed summary of that record *

Date	In (am)	Out (am)	Duration of Lunch	In (pm)	Out (pm)	Duration of Work	Overtime	Leave Type	Leave Used
Thu, Jan 1, 04	0:00	0:00	0:00	0:00	0:00	0:00	0:00		0:00
Fri, Jan 2, 04	7:00	11:30	0:30	12:00	15:30	8:00	0:00		0:00
Sat, Jan 3, 04	0:00	0:00	0:00	0:00	0:00	0:00	0:00		0:00
Sun, Jan 4, 04	0:00	0:00	0:00	0:00	0:00	0:00	0:00		0:00
Mon, Jan 5, 04	7:00	11:30	**1:00**	12:30	15:30	**7:30**	0:00		0:00
Tue, Jan 6, 04	7:00	11:30	0:30	12:00	15:30	8:00	0:00		0:00
Wed, Jan 7, 04	7:00	11:30	0:30	12:00	15:30	8:00	0:00		0:00
Thu, Jan 8, 04	7:00	11:30	0:30	12:00	13:00	**5:30**	0:00		2:30
Fri, Jan 9, 04	7:00	11:30	0:30	12:00	15:30	8:00	0:00		0:00
Sat, Jan 10, 04	9:00	11:30	0:00	0:00	0:00	2:30	**2:30**		0:00
Sun, Jan 11, 04	0:00	0:00	0:00	0:00	0:00	0:00	0:00		0:00

Total Hours: **Billable:** ***2046.5*** **Overtime:** ***3.5*** **Sick Leave Used:** ***10.5*** **Personal Leave Used:** ***4.5*** **Vacation Leave Used:** ***8***

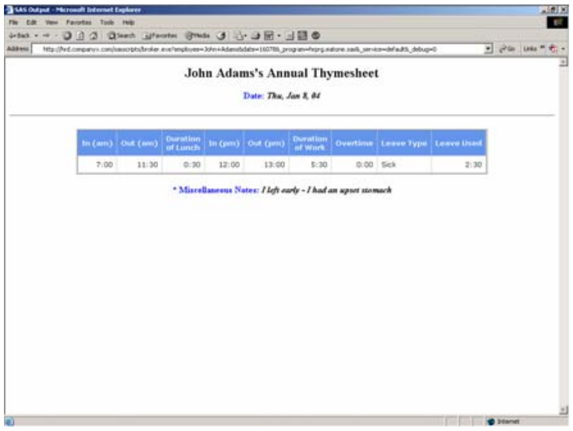
Figure 8. Modified output with formatted data using temporary variables where the date values become hyperlinks that call upon another SAS program used to generate a detailed summary report for a particular day selected by the user

When a user clicks on one of the links, a different SAS program (i.e. *eatone.sas*) gets invoked. The resulting report, a complete time schedule for the particular day that has been chosen, is displayed in the user's Internet browser:

Employee Annual Thymesheet Electronic Reporting System: Back-end (eatone.sas)

```
%macro eatone ;
  /* Output the title */
  data _null_ ;
    file _webout ;
    userdate=&date ;
    put "<H2 ALIGN=center>&employee's Annual Thymesheet</H2>" ;
    put '<H4 ALIGN=center><FONT COLOR=blue>Date: </FONT><I>'
        userdate weekdate15. '</I></H4><HR>' ;
  run ;
  /* Output the 1-day summary for the user-selected employee and date */
  ods html body=_webout style=sasweb ;
  proc print data=hrdat.timesheets noobs label ;
    where employee="&employee" and date=&date ;
    var inam outam lunch inpm outpm work overtime leavetype leaveused ;
    label inam='In (am)'
          outam='Out (am)'
          lunch='Duration of Lunch'
          inpm='In (pm)'
          outpm='Out (pm)'
          work='Duration of Work'
          overtime='Overtime'
          leavetype='Leave Type'
          leaveused='Leave Used' ;
    format inam outam lunch inpm outpm work overtime leaveused hhmm. ;
  run ;
  ods html close ;
  /* Output holiday and/or notes (if either one is not empty) */
  data _null_ ;
    file _webout ;
    set hrdat.timesheets ;
    where employee="&employee" and date=&date ;
    if holiday ne ' ' then put '<H4><CENTER><FONT COLOR=blue> *
      National Holiday: </FONT><I>' holiday '</I></CENTER></H4>' ;
    if notes ne ' ' then put '<H4><CENTER><FONT COLOR=blue> *
      Miscellaneous Notes: </FONT><I>' notes '</I></CENTER></H4>' ;
  run ;
%mend ;
%eatone ;
```

→



SAS Output - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://hrd.companyx.com/sasscripts/broker.exe?employee=John+Adams&date=16078&_program=hrprg.eatone.sas&_service=default&_debug=0
Go Links
Internet

John Adams's Annual Thymesheet

Date: *Thu, Jan 8, 04*

In (am)	Out (am)	Duration of Lunch	In (pm)	Out (pm)	Duration of Work	Overtime	Leave Type	Leave Used
7:00	11:30	0:30	12:00	13:00	5:30	0:00	Sick	2:30

* Miscellaneous Notes: *I left early - I had an upset stomach*

Figure 9. Detailed output for a particular day

Some managers tend to scrutinize the "excuses" certain employees have for arriving late, taking an extended lunch break, or leaving work early to go home. In order to compare the various notes employees have marked in their time schedules, these managers will open up multiple browsers, one for each day, and arrange them all on their screen. Considering the fact that it takes a lot of processing time to reload a complete time schedule report for each separate browser they need to open, the users have suggested that the detailed records appear in pop-up windows instead. That way, they only have to load the complete annual time schedule once to view multiple single-day observations.

In the same manner that HTML can be sent to an Internet source file using PUT statements in a DATA step, JavaScript can also be written out to better control program output. This scripting language can be used to create dynamic responses, modify the contents of HTML elements, or validate form data before it gets submitted to the server. For security reasons, Company X may require all its employees to log out of the internal Web site because some of the online reports contain sensitive data. JavaScript can then be added to the HTML output of every SAS/IntrNet program to display blinking text in the status bar at the bottom of users' Web browsers as a reminder:

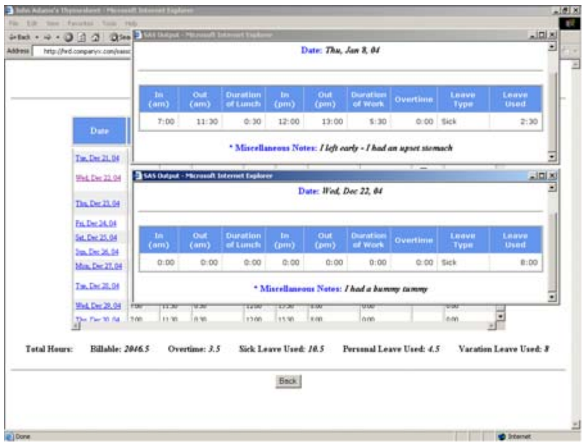
```
data _null_ ;
  file _webout ;
  put '<SCRIPT LANGUAGE="JavaScript">' ;
  put 'var blinkrate=250 ;' ;
  put 'var text="Remember to log out before leaving this terminal!!!" ;' ;
  put 'function showText() {' ;
  put 'window.status=text ;' ;
  put 'setTimeout("hideText()", blinkrate) ; }' ;
  put 'function hideText() {' ;
  put 'window.status="" ;' ;
  put 'setTimeout("showText()", blinkrate) ; }' ;
  put 'showText() ;' ;
  put '</SCRIPT>' ;
run ;
ods html body=_webout ;
proc print data=hrdat.passwords ;  run ;
ods html close ;
```

To deal with the managers' need to compare multiple records from a given time schedule all at once, JavaScript can be added to open up each single-day report in a different pop-up window. This JavaScript function will be placed with the HTML output using PUT statements in a DATA step. The unique hyperlinks that had been used to directly invoke the program to generate the detailed reports will instead call upon the new JavaScript function when clicked:

<u>Employee Annual Thymesheet Electronic Reporting System: Back-end (eatall.sas)</u>

```
%macro eatall ;
  /* Format date with hyperlinks that call a JavaScript function */
  data _null_ ;
    file _webout ;
    set hrdat.timesheets (end=last) ;
    tmpdate='<a HREF="javascript:popUp('||"'"||trim(employee)||
    "','"||trim(left(date))||')">'||put(date,weekdate15.)||'</A>' ;
    /** Conditionally format the fields as before **/
    if _n_=1 then do ;
        /** Build the headers as before **/
        put '<FONT COLOR=red><I>* Click on a date to view
              a detailed summary of that record *</I></FONT>' ;
        /** Open the tables as before **/
    end ;
    if mod(_n_,2)=1 then put '<TR ID=RowA>' ;
    else put '<TR ID=RowB>' ;
    put '<TD>' tmpdate '</TD><TD>' inam '</TD><TD>' outam
        '</TD><TD>' tmplunch '</TD><TD>' inpm '</TD><TD>'
        outpm '</TD><TD>' tmpwork '</TD><TD>' tmpovertime
        '</TD><TD>' tmpleavetype '</TD><TD>' leaveused
        '</TD><TD>' holiday '</TD><TD>' notes '</TD></TR>' ;
    if last then do ;
        /** Close the tables and build the footers as before **/
        put '<SCRIPT>' ;
        put 'function popUp(person, day) {' ;
        put 'var url="broker.exe?employee="+person+"&date="
              +day+"&_program =hrprg.eatone.sas&_service=
              default&_debug=0" ;' ;
        put 'winpops=window.open(url,"","height=250,width=750,
               scrollbars=yes,resizable=yes") ;' ;
        put '}' ;
        put '</SCRIPT>' ;
    end ;
    format inam outam inpm outpm leaveused hhmm. ;
  run ;
  /** Calculate and output the totals in the same manner as before **/
%mend ;
%eatall ;
```

→



Date: Thu, Jan 8, 04

In (am)	Out (am)	Duration of Lunch	In (pm)	Out (pm)	Duration of Work	Overtime	Leave Type	Leave Used
7:00	11:30	0:30	12:00	13:00	5:30	0:00	Sick	2:30

* Miscellaneous Notes: *I left early - I had an upset stomach*

Date: Wed, Dec 22, 04

In (am)	Out (am)	Duration of Lunch	In (pm)	Out (pm)	Duration of Work	Overtime	Leave Type	Leave Used
0:00	0:00	0:00	0:00	0:00	0:00	0:00	Sick	8:00

* Miscellaneous Notes: *I had a bummy tummy*

Total Hours: Billable: *2046.5* Overtime: *3.5* Sick Leave Used: *10.5* Personal Leave Used: *4.5* Vacation Leave Used: *8*

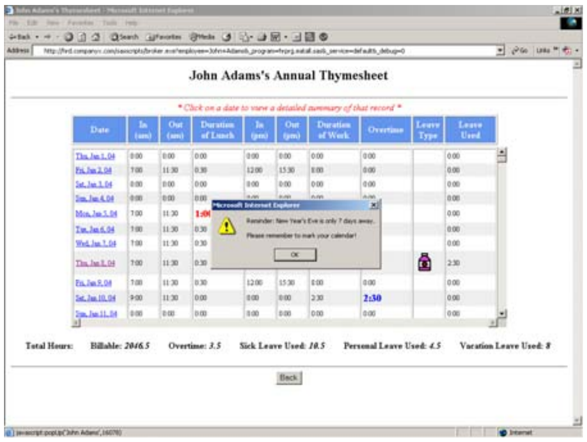
Figure 10. Detailed output for multiple days displayed in pop-ups

Any kind of JavaScript function can be added to output of a SAS/IntrNet application in this fashion. Accordingly, programmers may take advantage of JavaScript *confirm*, *prompt*, and *alert* dialog boxes to report certain messages to users. This is an effective way to save space while still providing users with an ability to view noteworthy content. An alert box reminding Company X staff members of upcoming holidays has been added to the EATERS using this idea. If a holiday is celebrated within the same week a time schedule report has been generated, an alert box informing the user of that information will then be displayed on the screen when the page has finished loading:

<u>Employee Annual Thymesheet Electronic Reporting System: Back-end (eatall.sas)</u>

```
%macro eatall ;

  /** Create and output the time schedule in the same manner as before **/

  /** Calculate and output the totals in the same manner as before **/

  /* Display a JavaScript alert box if a holiday is soon approaching */
  /* This message should appear after the time schedule has loaded */
  data _null_ ;
      file _webout ;
      set hrdata.timesheets ;
      where employee="&employee" and date > today() and
              date <= today()+7 ;
      if holiday ne "" then do ;
        put '<SCRIPT>' ;
        put 'alert("Reminder: ' holiday ' is only ' _n_ ' days away.
              \n\nPlease remember to mark your calendar!") ;' ;
        put '</SCRIPT>' ;
      end ;
  run ;
%mend ;
%eatall ;
```

→



John Adams's Annual Thymesheet

* Click on a date to view a detailed summary of that record *

Reminder: New Year's Eve is only 7 days away.

Please remember to mark your calendar!

Total Hours: Billable: *2046.5* Overtime: *3.5* Sick Leave Used: *10.5* Personal Leave Used: *4.5* Vacation Leave Used: *8*

Figure 11. Modified output with a JavaScript alert box reminder

▸ *Issue 4*

The HTML front-end of this application contains a drop-down box which holds the name of every Company X employee. Keeping this selection menu up to date, however, is a burdensome task for the developers considering the fact that the list of active employees changes all the time. Consequently, these programmers have to update the HTML form on a regular basis so that staff members can generate time schedule reports for any active employee and also avoid selecting a name from the hard-coded list of someone who is no longer included in the background data.

Just as a report resulting from the execution of a SAS/IntrNet program can be targeted to a user's browser, so can a GUI. Following the abovementioned approach, developers can build a Web interface to replace any front-end *.html* document; essentially, all the HTML code originally used to fashion the GUI should be placed within a SAS program and written out to the Internet source file with PUT statements in a DATA step. Rather than referencing a static *.html* document, users can instead arrive at the GUI by invoking the new program with the appropriate URL. The HTML front-end will then be dynamically generated and rendered in the user's browser. For example, this SAS program can be invoked from a Web browser in place of its *.html* counterpart to get to the company's contact page:

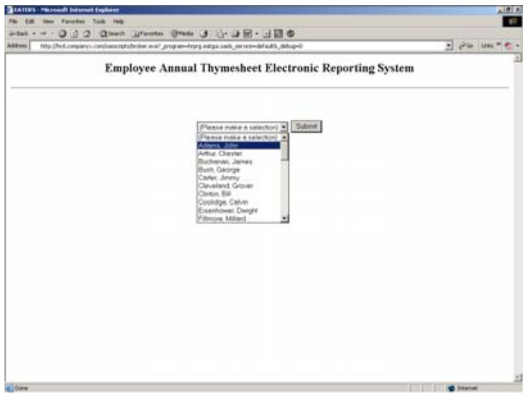
```
data _null_;
  file _webout ;
  put '<HTML>' ;
  put '<HEAD><TITLE>Contact Us</TITLE></HEAD>' ;
  put '<BODY>' ;
  put '<H2 ALIGN=center>Contact Us</H2><HR><BR><BR>' ;
  put '<CENTER><A HREF="mailto:support@companyx.com">Click here to send us a message</A></CENTER><BR><BR><HR><BR>' ;
  put '<CENTER><INPUT TYPE=button VALUE=Back onClick="javascript:history.back();"></CENTER>' ;
  put '</BODY>' ;
  put '</HTML>' ;
run ;
```

Users had formerly reached the EATERS front-end by typing the URL *http://hrd.companyx.com/eatgui.html* in their browser's address bar or clicking on a link that referenced this document. Now, users can enter *http://hrd.companyx.com/sasscripts/broker.exe?_program=hrprg.eatgui.sas&_service=default&_debug=0* from the Web address bar or simply click on the original link which has since been modified to point to this specific URL:

Employee Annual Thymesheet Electronic Reporting System: Front-end (eatgui.sas)

```
%macro eatgui ;

  /* Output the HTML code that make up the front-end GUI */
  data _null_ ;
    file _webout ;
    put '<HTML>' ;
    put '<HEAD>' ;
    put '<TITLE>EATERS</TITLE>' ;
    put '<SCRIPT>' ;
    /* Ensure that the user has made a selection before submission */
    put 'function chkForm() {' ;
    put 'if (document.employees.employee.selectedIndex == 0) {' ;
    put 'alert("Please make a selection.") ;' ;
    put 'return false ;' ;
    put '} else return true ;' ;
    put '}' ;
    put '</SCRIPT>' ;
    put '</HEAD>' ;
    put '<BODY>' ;
    put '<CENTER>' ;
    put '<H1>Employee Annual Thymesheet Electronic Reporting System</H1>' ;
    put '<HR><BR><BR><BR>' ;
    put '<FORM NAME=employees METHOD=get ACTION=
          "/sasscripts/broker.exe" onSubmit="return chkForm();">' ;
    put '<SELECT name=employee>' ;
    put '<OPTION VALUE=none SELECTED>(Please make a selection)' ;
    put '<OPTION VALUE="John Adams">Adams, John' ;
    put '<OPTION VALUE="Chester Arthur">Arthur, Chester' ;
    …
    put '<OPTION VALUE="George Washington">Washington, George' ;
    put '<OPTION VALUE="Woodrow Wilson">Wilson, Woodrow' ;
    put '</SELECT>' ;
    put '<INPUT TYPE=submit VALUE=Submit>' ;
    put '<INPUT TYPE=hidden NAME=_program VALUE=hrprg.eatall.sas>' ;
    put '<INPUT TYPE=hidden NAME=_service VALUE=default>' ;
    put '<INPUT TYPE=hidden NAME=_debug VALUE=0>' ;
    put '</FORM>' ;
    put '</CENTER>' ;
    put '</BODY>' ;
    put '</HTML>' ;
  run ;

%mend ;
%eatgui ;
```

→


EATERS - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address
Go Links

Employee Annual Thymesheet Electronic Reporting System

(Please make a selection) Submit
(Please make a selection)
Adams, John
Arthur, Chester
Buchanan, James
Bush, George
Carter, Jimmy
Cleveland, Grover
Clinton, Bill
Coolidge, Calvin
Eisenhower, Dwight
Fillmore, Millard

Figure 12. Modified GUI using a SAS program to dynamically generate the same HTML code that had been applied in the original *eatgui.html* document

These results are equivalent to those shown in Figure 1 (except for the updated title)

It is easy to dynamically build a front-end GUI by placing code from an *.html* document in a SAS program and having it written out to the Web on each invocation. This technique is especially valuable when certain HTML form fields need to be consistent with the data used in the background. One form field in particular that should parallel the back-end data is an HTML selection box. If new values are ever added or old ones removed from the underlying data, these changes should be made to the front-end selection menu(s) as well. Similar to the approach described earlier for writing out an HTML table to the Web, a selection box can be populated with data set values in a DATA step. In general, a selection box should be opened and closed on the first and last pass of the DATA step and each option should be written out for each iteration. In the case that the records are not all unique, the data set can be sorted beforehand with the *nodupkey* option so that the duplicate values will not be repeated in the selection menu:

```
proc sort data=dataset out=tmpdataset nodupkey ;  by variable ;  run ;
data _null_ ;
  file file ;
  set tmpdataset (end=last) ;
  if _n_=1 then put '<SELECT NAME=name>' ;
  put '<OPTION VALUE="' variable '">' variable ;
  if last then put '</SELECT>' ;
run ;
```

To avoid building a timesheet for an individual who is no longer working at Company X and ensure that a report can be generated for each new employee, the selection box on the front-end must contain only the names of all active staff members. Accordingly, the SAS code should be modified so that an option gets added to the drop-down menu for each unique name stored in the background data set. Since a new instance of this program runs each time a user opens the EATERS, the employees listed in the selection box will always mirror whatever is stored in the underlying data. Thus, the programmers do not have to be mindful of employee turnover as names are no longer hard-coded:

Employee Annual Thymesheet Electronic Reporting System: Front-end (eatgui.sas)

```
%macro eatgui ;

  /* Sort so names will be listed in alphabetical order and without duplication */
  proc sort data=hrdat.timesheets out=allnames nodupkey ; by employee ; run ;

  /* Output the HTML code that make up the GUI */
  data _null_ ;
    file _webout ;
    set allnames (end=last) ;
    if _n_=1 then do ;
      put '<HTML>' ;
      put '<HEAD>' ;
      put '<TITLE>EATERS</TITLE>' ;
      put '<SCRIPT>' ;
      /* Ensure that the user has made a selection before submission */
      put 'function chkForm() {' ;
      put 'if (document.employees.employee.selectedIndex == 0) {' ;
      put 'alert("Please make a selection.") ;' ;
      put 'return false ;' ;
      put '} else return true ;' ;
      put '}' ;
      put '</SCRIPT>' ;
      put '</HEAD>' ;
      put '<BODY>' ;
      put '<CENTER>' ;
      put '<H1>Employee Annual Thymesheet Electronic Reporting System</H1>' ;
      put '<HR><BR><BR><BR>' ;
      put '<FORM NAME=employees METHOD=get ACTION=
          "/sasscripts/broker.exe" onSubmit="return chkForm();">' ;
      put '<SELECT name=employee>' ;
      put '<OPTION VALUE=none SELECTED>(Please make a selection)' ;
    end ;
    put '<OPTION VALUE="' employee '">' employee ;
    if last then do ;
      put '</SELECT>' ;
      put '<INPUT TYPE=submit VALUE=Submit>' ;
      put '<INPUT TYPE=hidden NAME=_program VALUE=hrprg.eatall.sas>' ;
      put '<INPUT TYPE=hidden NAME=_service VALUE=default>' ;
      put '<INPUT TYPE=hidden NAME=_debug VALUE=0>' ;
      put '</FORM>' ;
      put '</CENTER>' ;
      put '</BODY>' ;
      put '</HTML>' ;
    end ;
  run ;

%mend ;
%eatgui ;
```

→

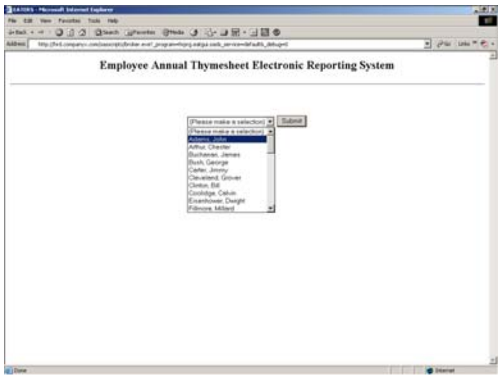

Employee Annual Thymesheet Electronic Reporting System

Figure 13. Modified GUI using a SAS program to dynamically generate the same HTML code that had been applied in the original *eatgui.html* document, but here the selection box is populated with the background data

These results are equivalent to those shown in Figure 12

It should be noted that there are alternative methods, such as *htmSQL*, that work to dynamically populate HTML form fields from the front-end; though, these tools require some additional setup and other programming knowledge.

Issue 5

The developers no longer have to regularly maintain a front-end portion of the EATERS given that the interface is dynamically generated with the underlying data used to populate the drop-down menu of all active employees. Still, these programmers find it to be a painstaking task to have to maintain 3 separate SAS programs, namely, *eatgui.sas*, *eatall.sas*, and *eatone.sas*, for just one SAS/IntrNet application. If there are ever any modifications that need to be made to one of the programs, it is usually the case that all three have to be revised to account for such changes.

It is possible to combine each portion of a SAS/IntrNet application into a single SAS program. Once the front-end *.html* document has been transformed into a SAS program by following the techniques described above, this piece can be conveniently integrated within the same program that is used to build the resulting Web reports. After each function is separated with if-then-else macro logic, a hidden flag variable (e.g. *mode*) can be passed along with every Internet request to control which of the two conditions to execute. The program *proceeds.sas*, shown below, is used to generate both the front-end Web interface and output reports that illustrate the daily profits earned at Company X:

```
/* The URL first called is http://hrd.companyx.com/sasscripts/broker.exe?_program=hrprg.proceeds.sas&_service=default&_debug=0&mode=gui */
%macro proceeds ;

  /* When mode=gui, a front-end interface containing a drop-down menu of each business day is displayed in the user's Web browser */
  %if &mode=gui %then %do ;
    data _null_ ;
      file _webout ;
      put '<HTML>' ;
      put '<HEAD><TITLE>Daily Proceeds</TITLE></HEAD>' ;
      put '<BODY>' ;
      put '<H2 ALIGN=center>Daily Proceeds</H2><HR>' ;
      put '<CENTER><BR><BR><FORM NAME=sales METHOD=get ACTION="/sasscripts/broker.exe">' ;
      put '<B>Select a day to create a proceeds report:</B>' ;
      put '<SELECT NAME=day>' ;
      put '<OPTION VALUE=Mon>Monday' ;
      put '<OPTION VALUE=Tue>Tuesday' ;
      put '<OPTION VALUE=Wed>Wednesday' ;
      put '<OPTION VALUE=Thu>Thursday' ;
      put '<OPTION VALUE=Fri>Friday' ;
      put '</SELECT>' ;
      put '<INPUT TYPE=submit VALUE=Submit>' ;
      put '<INPUT TYPE=hidden NAME=_program VALUE=hrprg.proceeds.sas>' ;  /* Call this same program when the form is submitted */
      put '<INPUT TYPE=hidden NAME=_service VALUE=default>' ;
      put '<INPUT TYPE=hidden NAME=_debug VALUE=0>' ;
      put '<INPUT TYPE=hidden NAME=mode VALUE=report>' ;  /* Set mode=report when the form is submitted to execute the other condition */
      put '</FORM><BR><HR><INPUT TYPE=button VALUE=Back onClick="javascript:history.back();"></CENTER>' ;
      put '</BODY>' ;
      put '</HTML>' ;
    run ;
  %end ;

  /* When mode=report, the resulting output report of daily proceeds is displayed in the user's Web browser */
  %else %if &mode=report %then %do ;
    %let fn='<INPUT TYPE=button VALUE=Back onClick="javascript:history.back();">' ;
    title "Proceeds report for &day" ;
    ods html body=_webout style=statdoc ;
    proc print data=hrdat.profits_&day noobs ;  footnote &fn ;  run ;
    ods html close ;
  %end ;

%mend ;
%proceeds ;
```

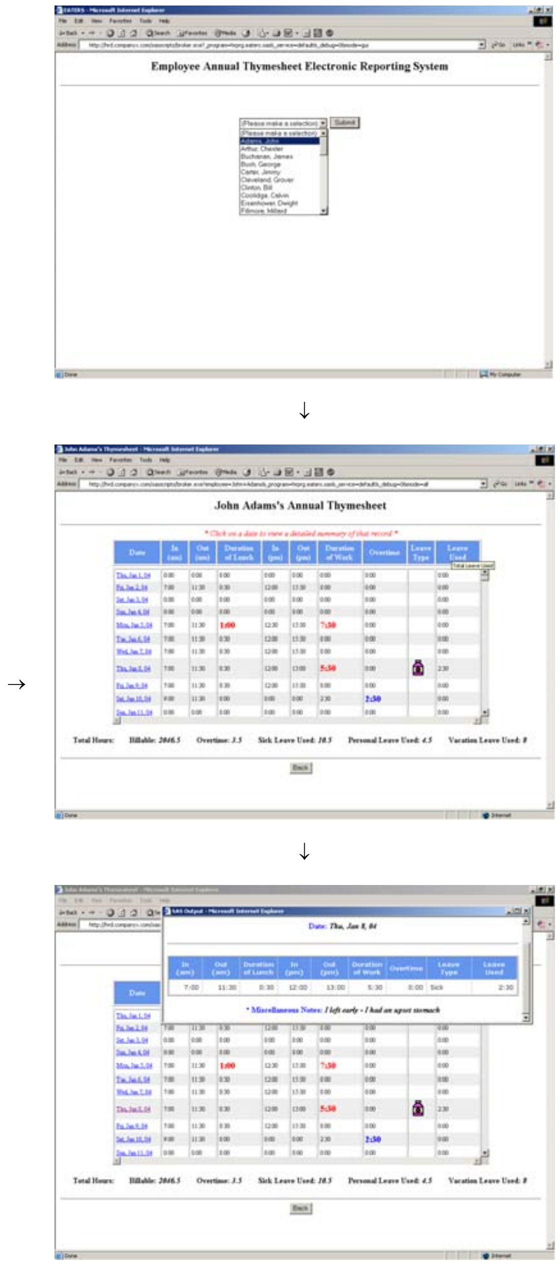
In effect, developers can add any number of unique functions within the same SAS program, yet only execute a particular one (or perhaps more than one) by passing in an appropriately assigned flag variable for each new request:

```
/* Execute a particular function based on the value of the passed flag variable mode */
%macro macro ;

  /* When mode is assigned the value function1, then perform this particular routine */
  %if &mode=function1 %then %do ;
    /** Insert code here for function1 **/
  %end ;

  /* When mode is assigned the value function2, then perform this particular routine */
  %else %if &mode=function2 %then %do ;
    /** Insert code here for function2 **/
  %end ;

  …

  /* When mode is assigned the value functionN, then perform this particular routine */
  %else %if &mode=functionN %then %do ;
    /** Insert code here for functionN **/
  %end ;

%mend ;
%macro ;
```

For this case, the developers would have a much easier time maintaining the EATERS if all 3 programs were merged together (i.e. *eaters.sas*), as opposed to having separate SAS programs to create the GUI (i.e. *eatgui.sas*), build the complete time schedules (i.e. *eatall.sas*), and generate detailed reports for particular days (i.e. *eatone.sas*). The flag variable *mode* should then be passed as a hidden parameter with each new Internet request to control which condition to execute. This parameter can be assigned the value *gui*, *all*, or *one*, depending on whether the user has chosen to arrive at the interface which holds the selection box listing all active employees, produce a time schedule report for a given staff member, or view the detailed summary information for a specific day of work, respectively:

Employee Annual Thymesheet Electronic Reporting System: Front- and Back-end (eaters.sas)

```
%macro eaters ;
  /* When mode=gui, output the HTML code that make up the front-end */
  /* _program=hrprg.eaters.sas&_service=default&_debug=0&mode=gui */
  %if &mode=gui %then %do ;
    /** Sort the data set to remove any duplicates in the same manner as before **/
    /* Output a form with a selection box holding the names of active employees */
    data _null_ ;
      file _webout ;
      set hrdat.timsheets (end=last) ;
      if _n_=1 then do ;
        /** Build the JavaScript and the header info, and open the form as before **/
        put '<SELECT name=employee>' ;
        put '<OPTION VALUE=none SELECTED>(Please make a selection)' ;
      end ;
      put '<OPTION VALUE="' employee '">' employee ;
      if last then do ;
        put '</SELECT>' ;
        put '<INPUT TYPE=submit VALUE=Submit>' ;
        put '<INPUT TYPE=hidden NAME=_program VALUE=hrprg.eaters.sas>' ;
        put '<INPUT TYPE=hidden NAME=_service VALUE=default>' ;
        put '<INPUT TYPE=hidden NAME=_debug VALUE=0>' ;
        put '<INPUT TYPE=hidden NAME=mode VALUE=all>' ;
        put '</FORM>' ;
        put '</CENTER>' ;
        put '</BODY>' ;
        put '</HTML>' ;
      end ;
    run ;
  %end ;
  /* When mode=all, output the annual timesheet for the selected employee */
  /* _program=hrprg.eaters.sas&_service=default&_debug=0&mode=all */
  %else %if &mode=all %then %do ;
    /* Output the employee’s timesheet in a scrollable region with embedded links */
    data _null_ ;
      file _webout ;
      set hrdat.timesheets (end=last) ;
      tmpdate='<a HREF="javascript:popUp('||"'"||trim(employee)||
      "','"||trim(left(date))||"')"'>'||put(date,weekdate15.)||'</A>' ;
      /** Conditionally format the fields as before **/
      if _n_=1 then do ;
          /** Build the headers and open the tables as before **/
      end ;
      /** Build each row of the time schedule as before **/
      if last then do ;
        /** Close the tables and build the footers as before **/
        put '<SCRIPT>' ;
        put 'function popUp(person, day) {' ;
        put 'var url="broker.exe?employee="+person+"&date="
              +day+"&_program=hrprg.eaters.sas&_service=
              default&_debug=0&mode=one" ;' ;
        put 'winpops=window.open(url,"","height=250,width=750,
              scrollbars=yes,resizable=yes") ;' ;
        put '}' ;
        put '</SCRIPT>' ;
      end ;
      format inam outam inpm outpm leaveused hhmm. ;
    run ;
    /** Calculate and output the totals in the same manner as before **/
    /** Display an alert if a holiday is approaching in the same manner as before **/
  %end ;
  /* When mode=one, output a detailed record summary for the selected day */
  /* _program=hrprg.eaters.sas&_service=default&_debug=0&mode=one */
  %else %if &mode=one %then %do ;
    /** Output a detailed record for a specific day in the same manner as before **/
  %end ;
%mend ;
%eaters ;
```



Employee Annual Thymesheet Electronic Reporting System

(Please make a selection)
Adams, John
Arthur, Chester
Buchanan, James
Bush, George
Carter, Jimmy
Cleveland, Grover
Clinton, Bill
Coolidge, Calvin
Eisenhower, Dwight
Fillmore, Millard

John Adams's Annual Thymesheet

* Click on a date to view a detailed summary of that record *

Date	In (am)	Out (am)	Duration of Lunch	In (pm)	Out (pm)	Duration of Work	Overtime	Leave Type	Leave Used
Thu, Jan 1, 04	0:00	0:00	0:00	0:00	0:00	0:00	0:00		0:00
Fri, Jan 2, 04	7:00	11:30	0:30	12:00	15:30	8:00	0:00		0:00
Sat, Jan 3, 04	0:00	0:00	0:00	0:00	0:00	0:00	0:00		0:00
Sun, Jan 4, 04	0:00	0:00	0:00	0:00	0:00	0:00	0:00		0:00
Mon, Jan 5, 04	7:00	11:30	1:00	12:30	15:30	7:30	0:00		0:00
Tue, Jan 6, 04	7:00	11:30	0:30	12:00	15:30	8:00	0:00		0:00
Wed, Jan 7, 04	7:00	11:30	0:30	12:00	15:30	8:00	0:00		0:00
Thu, Jan 8, 04	7:00	11:30	0:30	12:00	13:00	5:30	0:00		2:30
Fri, Jan 9, 04	7:00	11:30	0:30	12:00	15:30	8:00	0:00		0:00
Sat, Jan 10, 04	9:00	11:30	0:00	0:00	0:00	2:30	2:30		0:00
Sun, Jan 11, 04	0:00	0:00	0:00	0:00	0:00	0:00	0:00		0:00

Total Hours: Billable: *2046.5* Overtime: *3.5* Sick Leave Used: *10.5* Personal Leave Used: *4.5* Vacation Leave Used: *8*

Date: *Thu, Jan 8, 04*

In (am)	Out (am)	Duration of Lunch	In (pm)	Out (pm)	Duration of Work	Overtime	Leave Type	Leave Used
7:00	11:30	0:30	12:00	13:00	5:30	0:00	Sick	2:30

* Miscellaneous Notes: *I left early - I had an upset stomach*

Figure 14. A spiced up SAS/IntrNet application – the joint program *eaters.sas* is used to fashion the GUI, the annual time schedule reports, and the detailed daily summaries

CONCLUSION

SAS/IntrNet software provides developers with the facility to extend their SAS programs to the Web. With just some basic knowledge of HTML and JavaScript, programmers can create practical applications for producing, updating, and querying SAS data via the Internet. The paper "A Pinch of SAS®, a Fraction of HTML, and a Touch of JavaScript Serve Up a Grand Recipe" and its follow-up "A Recipe for Success: Migrating SAS/AF® Applications to the Web Using SAS®, HTML, and JavaScript," both presented uncomplicated ways of implementing SAS/IntrNet applications. This third installment offers additional insight on the subject by demonstrating a few programming techniques for spicing up such Web-based systems. Specifically, the content herein focuses on different strategies for resolving some of the usability and maintenance issues common among SAS applications that function online.

For the most part, the process of building a workable SAS/IntrNet application is straightforward; nonetheless, it can be challenging for programmers to develop Web reports that are graspable and appealing to end-users. Users have problems evaluating reports that are lengthy, crammed with information, and wasteful of space. Essentially, these Internet applications are not as effective when users are unable to easily sort through the data they require. One way of reducing this burden is by adding vertical scrollbars to excessively long tables. In this manner, users may be able to focus their attention on particular groups of records with greater ease and view summary information displayed at the bottom of such tables as soon as they load. For the cases where too much information is shown, programmers can conditionally format records with different fonts, sizes, and colors so that users may more easily and quickly identify noteworthy content. Expressive graphics can also be added within tables to give certain observations more emphasis and meaning. Using a similar technique, it is possible to embed hyperlinks within output tables that point to additional information. Accordingly, programmers can remove and replace supplementary information with links that reference other back-end programs, which may then be called upon to display the extra content as needed. All of these techniques can be applied in new and already existing SAS/IntrNet applications without much trouble.

SAS developers are faced with many challenges in designing Web-based systems that are more manageable on their end as well. With any Internet application, it is necessary for programmers to reliably update the front-end HTML form fields to match changes that occur in the background data. If these two components are not consistent, users may end up running queries that produce erroneous results and are wasteful of server-side resources. In addition, all the form field options users require to generate certain reports may not be available to them when these items do not reflect the data that has been changed. By replacing the hard-coded front-end *.html* document with a SAS program to dynamically generate the GUI for each request, such problems are eliminated as the form fields will always be populated using the underlying data. Thus, programmers no longer have to maintain the front-end portion of their SAS/IntrNet applications because the GUI will at all times be up to date. Using macro if-then-else conditional logic, developers can merge the front- and back-end processes of an Internet application into a single SAS program to foster a system that is easier to manage and always current. As users will still be able to transparently toggle between different functions, programmers will only have to maintain these separate routines in one joint program.

By applying the various techniques discussed, programmers can rework any SAS/IntrNet application to be more effective in operation. With such enhancements in place, users will have less difficulty examining tables online seeing that these Web reports will be more colorful, meaningful, and user-friendly. Developers will also benefit from these advancements as each separate component of a particular SAS/IntrNet application can be integrated into a single SAS program, one that requires very little maintenance. Ultimately, the different techniques shown provide for an application design that caters to the usability and efficiency needs of all end-users and programmers alike.

RESOURCES

Turner, Jonah P. "A Pinch of SAS®, a Fraction of HTML, and a Touch of JavaScript Serve Up a Grand Recipe," *Proceedings of the Twenty-Eighth Annual SAS® Users Group International Conference*, Paper 34-28, 2003.

Turner, Jonah P. "A Recipe for Success: Migrating SAS/AF® Applications to the Web Using SAS®, HTML, and JavaScript," *Proceedings of the Twenty-Ninth Annual SAS® Users Group International Conference*, Paper 25-29, 2004.

CONTACT INFORMATION

Jonah P. Turner
United States Census Bureau
4700 Silver Hill Road, Mail Stop 7500
Washington, DC 20233-7500
(301) 763-5420 or jonah.p.turner@census.gov