



The Missing Semicolon™

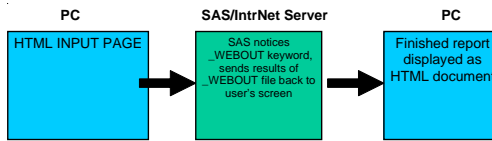
TECHNICAL ASSISTANCE FOR THE SAS® SOFTWARE PROFESSIONAL

Volume 8, Number 2

Summer 2006

SAS/IntrNet & Javascript

SAS/Intrnet allows end users to run SAS programs through a web interface and can be a perfect tool for non-SAS users. The interface web page created by a programmer for the end-user has parameters set up in HTML that the end user passes to the SAS program to customize the results. The simplicity of HTML, however, limits the user-friendliness of SAS/IntrNet interfaces and the complexity of the reports they can generate. By using JavaScript along with SAS, a programmer will have more flexibility when creating reports and interfaces. In particular, one of our favorite uses of JavaScript is to create dynamic drop-down menus based on information in SAS datasets.



For example, a user might be presented with an HTML page that has one text box saying “What variable do you wish to see?” The user might type in “ACCTNUMBER” and click SUBMIT. The value “ACCTNUMBER” is passed to the SAS program as a macro variable by the SAS/Intrnet Server. The program would run and the results are passed back to the web browser.

Macro variables passed through a web interface can be used by SAS programs for a variety of functions, some of which include:

- Selecting report or program to run
- Subsetting data
- Customizing look of report
- Entering information into a dataset

The results, in HTML format, are usually returned back to the browser of the person who submitted the reports. In the SAS code, this can be specified in two different places: either on the FILE= option on the ODS HTML statement or on the FILE= statement in a data _NULL_ step if the programmer has hard-coded the HTML statements directly into the program.

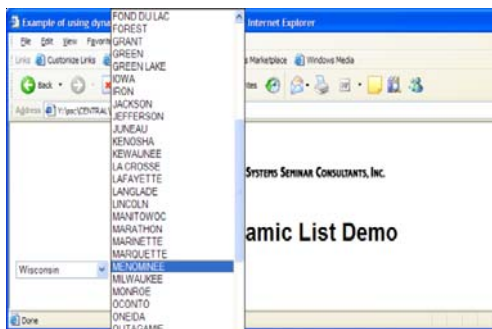
An example of a report generated with ODS, HTML and SAS/Intrnet might look like:

```

ODS HTML FILE=_WEBOUT;
PROC PRINT DATA=MYDATA.TEST;
  WHERE STATE='&MYSTATE';
RUN;
ODS HTML CLOSE;
  
```

Instead of using ODS to create an output file, an HTML file can be generated with PUT statements typed into a text editor. The completed document can be sent to the user's browser by using the same _WEBOUT keyword used in the ODS

Continued on page 2.....



Before we look at the code necessary to create dynamic drop-down menus, an overview will be given on SAS/Intrnet and JavaScript.

Quick Overview of SAS/IntrNet

SAS/IntrNet allows people who are unfamiliar with SAS to run report programs and view the results in their web browser. SAS/IntrNet runs on a server which awaits instructions from a user indicating which SAS programs to run, what options to use, and where the results should appear (typically back in the user's browser). The most common interface for SAS/IntrNet is a static HTML file with different input options in the form of text boxes, radio buttons, drop-down menus, etc. The values are passed to the SAS program that is run on the server as macro variables. The resulting report usually returns to the user's browser.



SYSTEMS SEMINAR CONSULTANTS, INC.

2997 Yarmouth Greenway Drive
Madison, WI 53711

www.sys-seminar.com
train@sys-seminar.com
1-800-997-7081

IN THIS ISSUE

SAS/Intrnet & Javascript
Alan Maas.....Page 1

President's Letter.....
Steve First.....Page 2

Open Positions.....
.....Page 2

Quick Tip.....
Sarah Chronquist.....Page 3

SAS/Intrnet Class.....
.....Page 4

Quick Tip.....
Tom Miron.....Page 5

Dear Ms. Sassy.....
.....Page 6

Training At Your Business Location.....
.....Page 6

Cognos Consulting.....
.....Page 7

Technical Credit and Recognition.....Page 8

SAS Training Solutions.....
.....Page 8



Letter From the President

SAS/IntrNet & Javascript

(CONTINUED FROM PAGE 1)



Dear SAS User:

After several sluggish years, the IT community is enjoying the best economy in several years. We have seen a great increase in open IT positions, contract work, and training.

In the SAS world, the traditional SAS applications are more vibrant and dependable than ever, and exciting new enhancements and products are becoming more and more popular.

SAS Business Intelligence is now one of the most popular BI products. The SAS developers assure us that they are well aware of the features of competitive BI systems, and they are working very hard to become the number one BI provider.

Systems Seminar Consultants has been working with clients to implement and optimized the SAS Business Intelligence Suite. We have also just added a new SAS Enterprise Guide class to our curriculum.

Enterprise Guide 4.0 allows the casual users to generate SAS code via wizards and menus, and it offers more functionality than ever before. Experienced SAS users can also use EG to alter generated code, or they can use it to write and alter SAS programs. EG gives welcome shortcuts that simplify management of data and libraries. Libraries, SAS datasets, Excel data, and raw data and can be very confusing to many new SAS users, and EG does a nice job of eliminating tedious coding for those new users. We are very excited about the addition of this innovative product to our training curriculum.

We enjoy helping clients get the most out of their SAS products, whether it be just Base SAS to the vertical BI line!



HTML statement. Using an input variable that we can now access as a macro variable, the following sample SAS program will generate the skeleton of an HTML document:

```
DATA _NULL_ ;
  FILE _WEBOUT ;
  PUT "<HTML>" ;
  PUT "<HEAD>" ;
  PUT "</HEAD>" ;
  PUT "<BODY>" ;
  PUT "The text you typed in was &MY_VAR" ;
  PUT "</BODY>" ;
  PUT "</HTML>" ;
RUN ;
```

This program returns an HTML document with SAS/IntrNet that displays "The text you typed in was ACCTNUMBER" after a user passed the value 'ACCTNUMBER' to the data_null_ step. Using the same basic idea, several macro variables can be sent to a program which uses the values of the variables to subset data or create interesting reports. For example, a user might type in the name of a variable in order to get its values or type in a minimum dollar account and get all accounts with at least that balance, or type in an ID number and get the associated customer information. While there are several solutions to most reporting needs, with HTML and JavaScript you can truly enhance your ability to create dynamic content for end-users.

Quick Overview of JavaScript

Technically speaking, JavaScript is a "cross-platform, object-oriented scripting language that can be interpreted by browsers while embedded in HTML code." More practically speaking, JavaScript allows you to treat most HTML content like objects in a more powerful programming language.

Open Positions

Looking for Well-Qualified candidates in:

- ◆ Madison
- ◆ Minneapolis
- ◆ Chicago

From **developer** to **programmer** to **analyst** opportunities, we will find the right fit for you.

Apply online at: <http://www.sys-seminar.com/onlineapp.php>



SYSTEMS SEMINAR CONSULTANTS, INC.

Copyright © 2006 Systems Seminar Consultants, Inc. Madison, WI. All rights reserved. Printed in USA.

The Missing Semicolon is a trademark of Systems Seminar Consultants, Inc. SAS, SAS/IntrNet, and SAS/ACCESS are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries.

Some of the most common uses of JavaScript are creating and manipulating variables, creating user defined functions which can be run on triggers (onClick, onMouseOver etc), using built in math and string functions, accessing arrays and using them in drop-down menus, and changing almost any option or setting of any HTML object. JavaScript also uses a great namespace convention.

Objects are referenced by parent.child.option-like statements. For example, first a form might be named, then an input variable and finally the option that can be changed. So myform.mypicture.src="picture2.jpg"; would change the mypicture object from whatever it was displaying to picture2.jpg. Even more simply, the keyword "this" can be used to reference an object the user is already "in". This is usually used within triggers, because a "this" reference makes sense in that context. For example, onFocus="this.value='Enter Name';" would allow a default value to show up when someone clicked on a text box in which they were supposed to enter their name. The internet is a great source for examples and tutorials. Several wonderful JavaScript sites that offer free tutorials and working examples of code are available.

There are three conventional ways to use JavaScript within HTML. The first is to use <script> "code" </script>. All code within the script tags will be interpreted as JavaScript code. To be safe, you can also use <script language=""></script> to specify a version of JavaScript. Another way to use JavaScript code is to use <script src="path/filename.js"></script>. This is equivalent to a %INCLUDE statement in SAS. The .js file is read in and all of its code is included in the HTML file. This can be used several times for several different JavaScript files. There is nothing special about the .js file. You can create one in notepad and save it with a .js extension. The final way to implement JavaScript code is through event handlers or "triggers". In an object, if you specify an event handler such as onClick="...stuff..." whatever is in the parentheses will automatically be executed as JavaScript code when the event is triggered. We will focus on the last two ways to use code in the example.

Creating Dynamic Drop-Down Menus

Most Internet users have probably used dynamic drop-down menus on a regular basis. Searching for a car? If you've used a menu that first allows you to select the manufacturer and then populates a second menu with that manufacturer's models you've used a dynamic drop-down menu.

Without JavaScript or another scripting language, we would get this functionality in SAS/Intrnet by either showing all car models on the second drop-down regardless of what manufacturer was selected in the first, or, more likely, having the selection broken out to two screens. The user would first enter the manufacturer and then hit submit. A second drop-down list would be returned from SAS (ased on values in a dataset, asking the user to next select the car model.

Just as SAS, through SAS/Intrnet, was used to create the second drop-down list, we can also use SAS to create a JavaScript file based on the data which will allow the user to make all their selections on one easy to understand form. By using SAS to create the .js file used in the SAS/IntrNet HTML reporting front-end, the interface dynamically

QUICK TIP

These features are new for SAS version 9. First, import an Excel spreadsheet with a libname statements and set the specific options. Then, point to workbook using Excel engine. You must have SAS/ACCESS Interface to PC Files licensed and installed at your site to use the Microsoft Excel libname engine.

```
LIBNAME myxls 'C:\temp\sales.xls';
```

```
GETNAMES = YES|NO
```

Determine whether SAS will use the first row of data in a Microsoft Excel worksheet or range as column names.

YES – use the first row of data or range as column names

NO – do not use the first row of data. SAS generates variable names: F1, F2, etc.

The default is YES.

```
USEDATE = YES|NO
```

This Specifies whether to use the DATE9. format for date/time values in Excel workbooks.

YES – date/time values assigned DATE9. format.

NO – date/time values assigned DATETIME. format.

The default is YES.

run;

changes the instant any data changes. Users will always have the most current selections available.

This example will show how to turn a SAS dataset with relational data into a dynamic drop-down menu that can be implemented in current reporting interfaces within SAS/IntrNet. Everything shown here can be contained within one SAS program that is run whenever the reporting interface is accessed. First the .js file needs to be created:

```
DATA MYDATA;  
  SET MYORIGINALDATA;  
  RENAME VARIABLE_X=PRIMARY;  
  RENAME VARIABLE_Y=SECONDARY;  
RUN;
```

```
PROC SQL NOPRINT;  
  SELECT COUNT(DISTINCT PRIMARY)+1 INTO :NUMBTYPES  
  FROM MYDATA;  
QUIT;
```

```
%LET NUMBTYPES=&NUMBTYPES;
```

```
PROC SORT DATA = MYDATA;  
  BY PRIMARY SECONDARY;  
RUN;
```

The variables used in creating the JavaScript and HTML files are aptly named primary and secondary (think of it as primary = manufacturer, secondary = model). Simply changing the set statement to include your own dataset and renaming the appropriate variables will allow you to use the rest of the code. The SQL statement retrieves the number

of different types (or manufacturers). Type is the higher level menu when it is referred to as such. For some of the loops just ahead, this value is important for setting up values of an array. Finally, the data is sorted so it looks ordered in the menus.

```
DATA _NULL_;
FILE JSOUT;
IF _N_=1 THEN
DO;
PUT " VAR GROUP= NEW ARRAY(&NUMBTYPES);";
PUT " FOR (I=0; I<&NUMBTYPES; I++);";
PUT " {";
PUT " GROUP[I]=NEW ARRAY();";
PUT " GROUP[I][0]=NEW OPTION('---PLEASE SELECT A
SECONDARY---', ' ');";
PUT " }";
```

The code above starts the .js file. The filename statement for JSOUT can point to any flat file, as long as you use the same path later when accessing the created file. First an array called group is created, with a first dimension size equal to the number of types in the data. This will store all of the related values. The loop then creates an array for each type that will hold each specific token (model). The second statement within the loop sets the first value of every token array as the default value "Please select a secondary." The IF _n_=1 piece of code ensures that the lines are input only once. If left off, the data step can create redundant code in the .js file. This method is used more than once in the example and is often the reason for non-intuitive END statements.

```
DO UNTIL (ENDLIST=1);
SET MYDATA END=ENDLIST;
BY PRIMARY;
IF FIRST.PRIMARY THEN N+1;
M+1;
PUT " GROUP[ " N " ][ " M " ]=NEW OPTION(" SECONDARY
" ', 'ANY VALUE');";
IF LAST.PRIMARY THEN M=0;
END;
END;
```

In this code, the values of our two-dimensional array are set. The variable n represents the position of the primary values and m represents the position of the secondary values. The new Option statement is a JavaScript keyword that creates a label and value in an array position. Thus, the value of the current secondary field will be what is actually displayed in the drop-down menu to the user, and 'any value' will be the value behind the scenes associated with that choice. This can be changed to anything, which is why it is hard-coded here as such. Perhaps a label of "less than 30,000" would have a value of "le 30000" to be used in reporting. This value is what is sent to SAS as a macro variable when another SAS program is run from these results.

```
IF _N_=1 THEN
DO;
PUT "FUNCTION INIT()";
PUT "{";
PUT " DOCUMENT.MAINFORM.PRIMARY.OPTIONS[0] = NEW
OPTION(' SELECT TYPE ', '0');";
END;
IF _N_ = 1 THEN
DO;
DO UNTIL (ENDLIST2=1);
SET MYDATA END=ENDLIST2;
BY PRIMARY;
```

```
IF FIRST.PRIMARY THEN
DO;
K+1;
PUT " DOCUMENT.MAINFORM.PRIMARY.OPTIONS[" K "] = NEW
OPTION(" PRIMARY " ', 'ANY VALUE');";
END;
END;
DO;
PUT " }";
```

Because the values on the first drop-down menu will never change in the session, those values can be coded based on the data right when the page loads. For that reason, the function is called init(). In the HTML page sent back to the user, init() is run "onLoad." This initializes all the values of the first drop-down menu to all distinct primary values in our dataset. The JavaScript code document.mainform.primary.options[n]= points to a drop-down menu called 'primary' within a form called 'main' within the document. The .options portion tells the drop-down menu that we are creating a new menu choice with the following label and value.

```
PUT " function populate(x)";
PUT " {";
PUT " var formfield=document.mainform.secondary;";
PUT " for(m=formfield.options.length-1;m>0;m--);";
PUT " formfield.options[m]=null;";
PUT " for(i=0;i<group[x].length;i++);";
PUT " {";
PUT " formfield.options[i] = new Option
(group[x][i].text , group[x][i].value);";
PUT " }";
PUT " formfield.options[0].selected=true;";
PUT " }";
END;
```

SAS/Intrnet Class

This course will be customized for your

SAS®/Intrnet Environment

At Your Site

- ◆ Setup
- ◆ Convert a Program for Your Site

Introduction to

- ◆ SAS Macros
- ◆ HTML
- ◆ ODS

HTML

- ◆ Creating Forms
- ◆ HTML Output from SAS
- ◆ Exporting to Excel

Styles

- ◆ SAS[®] Styles
- ◆ Cascading Stylesheets
- ◆ Troubleshooting

Call 1-800-997-7081 to discuss details!

END;

The populate function does the grunt work of this process. As you will see in the HTML page, we code this function to run "onChange" within the primary drop-down menu. This means that any time the primary drop-down menu changes, populate is run. In other words, populate completely empties out the secondary drop-down menu and refills it with the corresponding values of the new primary selection. Thus, if the current primary selection is Toyota, the secondary menu will contain things like Corolla, Camry, etc. If you change the value of the first menu to Honda, populate() will empty out the secondary menu and fill it with Civic, Accord etc.

```

DATA _NULL_;
  FILE _WEBOUT;
  PUT "<HTML>";
  PUT "<HEAD>";
  PUT "<SCRIPT src='filepath/filename.js'
  language = 'JavaScript'> </SCRIPT>";
  PUT "</HEAD>";

```

```

PUT "<BODY onLoad='init();'>";
PUT "<FORM NAME='mainform'>"
PUT "<SELECT name='primary' onChange =
'populate(this.options.selectedIndex)'>";
PUT "</SELECT>";
PUT "<SELECT name='secondary'><option value='
' selected> Please select a primary first </
option>";
PUT "</SELECT>";
PUT "</FORM>";
PUT "</BODY>";
PUT "</HTML>";

```

RUN;
END;

Sample Results:

Dynamic List Demo

Dog	—Please select a secondary—
	—Please select a secondary—
	collie
	terrier
	pug
	poodle

Dynamic List Demo

Cat	—Please select a secondary—
	—Please select a secondary—
	siamese
	calico
	mixed

Coding the HTML is generally much easier than coding the JavaScript file. This code creates a page with two drop-down menus on it named primary and secondary. It calls a function init() when the page is loaded and it uses the .js file that was just created. When the first drop-down menu changes, it clears out the second menu and replaces it with new values associated with the current selection. The selectedIndex keyword is required to send a numeric value to the populate function to tell it the position of the currently selected member of the first drop-down menu. We now have a system in which one drop-down menu depends on the other for content. Once you understand and can replicate this code, it can be used anywhere in the same manner by changing the dataset that is accessed and renaming the appropriate variables to primary and secondary. Furthermore, a third menu contingent on the second can be created in the same way. Using JavaScript and SAS/Intrnet together makes for a flexible and fun reporting solution.



SAS® Consulting Services

Recently Purchased SAS Business Intelligence Products?

We Can Help with SAS BI!

- ◆ Architectural Design
- ◆ Administration
- ◆ New Application Development or Enhancements
- ◆ Training

Extensive SAS® Experience

- ◆ Data Systems Development
- ◆ Decision Support and Business Consultation
- ◆ Data Manipulation
- ◆ Report Design
- ◆ Development
- ◆ Web-enablement.
- ◆ Consulting Partners of the SAS® Institute.
- ◆ SAS Certified Professionals

New Cognos Consulting Partners

Support for Products Includes:

- ◆ PowerPlay
- ◆ ReportNet

Call 1-800-997-7081
for your Personalized Solution!

QUICK TIP

To enclose code or text within a block comment, highlight the selected code and press Ctrl+/

To remove a block comment, highlight the selected code and press Ctrl+Shift+/

This works in the SAS enhanced editor.



Dear Ms. Sassy...

Sorted Non-SAS Datasets

From time to time I am asked about sorting and efficiency, particularly when utilizing non-SAS datasets.

Dear Ms. Sassy,
I know sorting is inefficient. I have a program that reads a non-SAS data set. It is already in the sequence I need. I use a sort step so that SAS knows it is sorted. Is there some way to tell SAS that the data is already sorted so that I don't have to sort the data set?

Sincerely,
Out of Sorts

Dear Out of Sorts,
There is a data set option called SORTEDBY that you can use when creating your SAS data set. The example below has three steps.

The first step reads in the non-SAS data set and uses the SORTEDBY data set option on the DATA statement.

The second step is a PROC CONTENTS step. The output indicates that the data set IS sorted.

The third step is a PROC SORT step. The log indicates that the data set was not sorted because it is already in the sort sequence.

Note: if you do use the SORTEDBY option to specify the sequence, but the data set is NOT in that order, you will receive an out of sequence error in your log when you try to use it in a step that requires sorting.

LOG INFORMATION:

```
30 /* SORTEDBY' DATA SET OPTION */
31 OPTIONS NOCENTER;
32 TITLE 'SORTEDBY OPTION';
33
34 DATA TEMP
35     (SORTEDBY=PRIORITY
36     DESCENDING INDATE);
37     INPUT @1 PRIORITY 1.
38           @3 INDATE DATE7.
39           @11 OFFICE $2.
40           @14 CODE $3. ;
41     FORMAT INDATE DATE7.;
42     DATALINES;
```

NOTE: THE DATA SET WORK.TEMP HAS 8 OBSERVATIONS AND 4 VARIABLES.

NOTE: DATA STATEMENT USED:

REAL TIME 0.02 SECONDS

CPU TIME 0.02 SECONDS

```
51 ;
52 RUN;
```

```
54 PROC CONTENTS; RUN;
```

```
NOTE: PROCEDURE CONTENTS USED:
      REAL TIME          0.02 SECONDS
      CPU TIME           0.02 SECONDS
```

```
55 PROC SORT DATA=TEMP;
56     BY PRIORITY;
57 RUN;
```

NOTE: INPUT DATA SET IS ALREADY SORTED, NO SORTING DONE.

```
NOTE: PROCEDURE SORT USED:
      REAL TIME          0.00 SECONDS
      CPU TIME           0.00 SECONDS
```

```
SORTEDBY OPTION
                                14:10 TUESDAY, APRIL 18, 2006    2
THE CONTENTS PROCEDURE
DATA SET NAME:WORK.TEMP          OBSERVATIONS:              8
MEMBER TYPE: DATA              VARIABLES:                  4
Engine: V8                      Indexes:                0
CREATED: 14:10 TUESDAY, APRIL 18, 2006
OBSERVATION LENGTH: 24
```

SAS Training at Your Business Location

- ◆ Introduction to SAS®
- ◆ SAS® Report Writing
- ◆ Introduction to PROC Report
- ◆ The SAS® SQL Procedure
- ◆ Advanced SAS®
- ◆ Using SAS/ACCESS® with Relational Databases
- ◆ SAS® Efficiencies
- ◆ Mainframes Made Easy
- ◆ Advanced Macros
- ◆ Tips, Tricks, & SAS Techniques
- ◆ Exploiting the SAS® Output
- ◆ Delivery System
- ◆ SAS® Macros

View our course catalog at
www.sys-seminar.com/training.php

Call 1-800-997-7081 to discuss details!

***Recently Purchased
SAS Business Intelligence Products?***

We Can Help with SAS BI!

- ◆ Architectural Design
- ◆ Administration
- ◆ New Application Development or Enhancements
- ◆ Training

Call 1-800-997-7081 to discuss details!

```

LAST MODIFIED: 14:10 TUESDAY, APRIL 18, 2006
DELETED OBSERVATIONS: 0
PROTECTION:                                COMPRESSED:        NO
DATA SET TYPE:                               SORTED:              YES
LABEL:
  --ENGINE/HOST DEPENDENT INFORMATION--
DATA SET PAGE SIZE:                          4096
NUMBER OF DATA SET PAGES:                   1
FIRST DATA PAGE:                           1
MAX OBS PER PAGE:                           168
OBS IN FIRST DATA PAGE:                    8
NUMBER OF DATA SET REPAIRS:                0
FILE
/NAME:
C:\SAS TEMPORARY FILES\_TD2688\TEMP.SAS7BDAT
RELEASE CREATED:                             8.0202M0
HOST CREATED:                               WIN_PRO
--ALPHABETIC LIST OF VARIABLES AND ATTRIBUTES--
#    VARIABLE    TYPE    LEN    POS    FORMAT
-----
4    CODE        CHAR    3      18
2    INDATE     NUM     8      8      DATE7.
3    OFFICE     CHAR    2      16
1    PRIORITY   NUM     8      0

--SORT INFORMATION--
SORTEDBY:    PRIORITY DESCENDING INDATE
VALIDATED:   NO
CHARACTER SET: ANSI

```

Since the sorting of the data is unnecessary, you can leave the PROC SORT step out completely.

Sincerely,
Ms. Sassy

Dates and Substrings

I recently received an email from one of our former students:

Dear Ms. Sassy,
Does Substring work with date fields?
I have the following logic:

```

TODAY = DATE();
YR4 = YEAR(TODAY);
YR2 = SUBSTR(YR,4,3,2);
PUT _ALL_;

```

When I look at the PDV information, it shows YR4 = 2006, but YR 2= BLANK.

Sincerely,
Out of Date

Dear Out of Date,

You cannot use the SUBSTR function on numeric fields. Remember that numeric variables, including SAS date fields, are stored as 8-byte floating point, not text strings. A numeric variable needs to be converted to a character value by using the PUT function to use the SUBSTR function. Please note that YR4 is a numeric variable and YR2 is a character variable.

I am not sure how you intend to use the YR variable, but if you need the 2-character year, you can use the following code. The run log is shown below.

```

PROGRAM:

DATA _NULL_;
  TODAY = DATE();
  YR4 = YEAR(TODAY);
  YR2 = SUBSTR(PUT(YR4,Z4.),3,2);
  PUT _ALL_;

RUN;

LOG:
34  DATA _NULL_;
35  TODAY = DATE();
36  YR4 = YEAR(TODAY);
37  YR2 = SUBSTR(PUT(YR4,Z4.),3,2);
38  PUT _ALL_;
39
40  RUN;

```

TODAY=16891 YR4=2006 YR2=06 _ERROR_=0 _N_=1

Sincerely,
Ms. Sassy



Cognos Consulting Partners

Support for Products Includes:

◆ **ReportNet**

◆ **PowerPlay**

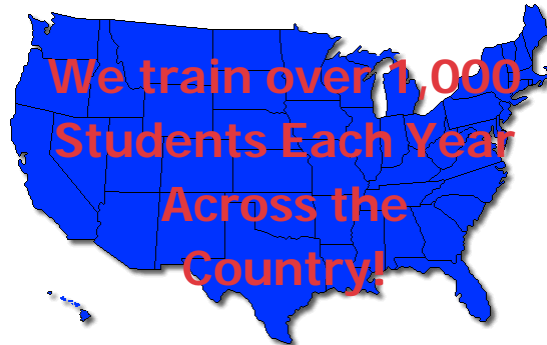
- ◆ Report Design
- ◆ Project Specifications
- ◆ Project Management
- ◆ Data Modeling
- ◆ Programming
- ◆ Report Creation
- ◆ Final Documentation
- ◆ End-User Training

Call 1-800-997-7081 to discuss details!

SAS Training Solutions at Your Business Location

Complimentary Follow-up Help Desk
Competitive Prices

Customized Materials & Courses
Quality Training **Nationwide**



New This Fall...
Enterprise Guide

Interested in training...Unsure of how to coordinate?

- ◆ Have your SAS users take our training survey at www.sys-seminar.com/sscsurvey.php
- ◆ We'll compile reports to **minimize expenses & maximize results**

View our Course Catalog at www.sys-seminar.com

Call 1-800-997-7081 ext. 306 to discover your SAS training solution.

TECHNICAL CREDIT

AND RECOGNITION



Steve First
President

Sarah Chronquist
Trainer/Consultant



Jennifer First
Editor



Katie Ronk
Director of
Operations

Rosalind Gusinow
Trainer/Consultant



Kathleen Nosal
Editor