Teaching SAS ® Web Publishing in a Web Environment

Roderick A. Rose, Jordan Institute for Families, UNC-Chapel Hill, North Carolina
Sally S. Muller, Jordan Institute for Families, UNC-Chapel Hill, North Carolina

ABSTRACT

Experienced SAS users gain a more comprehensive understanding of static SAS web publishing when they are taught in an interactive classroom setting with a web page as the primary instructional device. Web-based instruction goes beyond conventional instructional methods, such as overheads or computer-based slide presentations, offering a full range of instructional materials and references, and improved interactive demonstrations and self-tutorials of SAS web publishing.

The authors developed a website, on which HTML, FTP, DATA _NULL_, the HTML Formatting Macros and the Output Delivery System (HTML destination) are all seamlessly integrated into one interactive session. This website employs cascading style sheets and client-side Javascript, providing the instructor with a set of interactive elements using a fully portable instructional aid that permits the instruction on any computer with an Internet connection and browser. These elements include, but are not limited to: links that open a new browser for additional resources during the lecture and on subsequent review, examples of programs displayed side-by-side with their output, and downloadable programs and data sets. The instruction is based in program building, and not a reference. The intended audience of this presentation includes prospective SAS/IntrNet instructors.

CONCEPTS TO COVER

The process of developing static SAS-based output on the web involves a background in the client/server application and several iterative steps: HTML, SAS programming, uploading, and debugging. These steps, and the concepts associated with them, serve as the basis of our course.

THE CLIENT/SERVER APPLICATION

By introducing the SAS Web Formatting Tools, SAS has made it practical to store all applications and data on a central server and allow interaction with remote clients using a web browser. Thus, the first concept that the instructors address is the client/server application.

Client/servers applications encompass organization-wide Intranets as well as the Internet, and provide the pipeline over which all web pages are transmitted. Static web pages are created and stored on a web server for distribution. Minimal user interactivity with the page is possible - users can select links to move from one page to the next, for instance - but all elements displayed on a web page must be incorporated into the design of the page by the author before it is published. Still, we tend to think of web pages as "on-the-fly" creations, as authors can instantaneously and continuously update the page.

HTML

SAS programmers no longer have to HTML-format output – as far from “on-the-fly” as one can get – as it is now possible to create SAS output in a completely web-ready format. Using a text editor to draft a web page is still an essential skill. For this, one still needs a basic knowledge of HTML.

There are two reasons for providing a basic HTML instruction:

- To give the programmer an appreciation for how a web page is constructed.
- To provide some essential markup skills, which will complement the DATA _NULL_ learned further on.

SAS PROGRAMMING

The only prerequisite of the course is basic SAS programming knowledge. To the extent that this prerequisite is satisfied the instruction can focus on the SAS HTML Formatting Tools and the Output Delivery System:

- For the purpose of adding content like links and images to the page, the programmer requires the DATA _NULL_ step and PUT statements for writing HTML.
- The HTML Formatting Tools – macros - can display SAS data sets and all SAS procedural output and are very simple to invoke.
- The Output Delivery System is capable of producing all procedural output in a formatted style.

UPLOADING THE STATIC APPLICATION

The applications that are developed must be uploaded from the local (client) machine to the web server to be accessible on the Internet. The students will need to gain familiarity with another widely-used Internet protocol, the File Transfer Protocol (FTP).

FTP applications can differ widely. WS-FTP and the Windows 95/98 FTP Client are useful for general uploads. The SAS FTP Access Method, wherein the file transfer protocol is written directly into the SAS programs, is the preferred application, and is used in the class.

DEBUGGING

SAS-generated Internet applications are comprised of SAS and HTML-based components. Unfortunately, mistakes and bugs are comprised of both SAS and HTML-based components as well. The best method for debugging a SAS application is to study the log file; the best method for debugging HTML is to study the output. Many of the errors that can occur are specific to either SAS or HTML. However, there are problems, like unclosed quotes, that occur because of inherent incompatibilities. The trick is to approach the programming in an iterative fashion – design the HTML first, then build the SAS code around it. This does not prevent all problems from arising, but it does reduce them. The authors thus chose to emphasize the iterative approach to building static SAS applications.

CONSTRUCTING THE WEBSITE

The website consists of several layers of user interaction. The first, and most important layer, consists of a series of sequential pages that provide the foundation for the class. Additional layers allow the students to learn more about the concepts of greatest important to them, by providing demonstrations, links to other websites and further instruction.

The first section begins with an introduction to SAS web publishing, and tries to answer a basic question: What advantages accrue to the SAS user who can publish static online content “on-the-fly”?

The second section is devoted to generating basic online output and consists of three lessons:

- The first lesson is an instruction in basic HTML, in which elements, attributes and tags are defined and reviewed. After learning several simple but useful HTML elements, the students author a simple web page.
- The second lesson combines HTML and SAS code by showing the students how to construct a DATA _NULL_ step with PUT statements with which the SAS System will author the HTML for them. In particular, it will author the same page that they wrote in the first lesson.
- The final lesson of the section describes the SAS FTP Access Method. The students uploads their pages.

When the students complete this section, they have a home page accessible online which was authored by using the SAS system.

Macros are the focus of the third section. First, since no assumptions have been made about the students' understanding of macros, an introduction to macros is given. The remainder of the section consists of three lessons, each one designed around one of three HTML Formatting Macros:

- The Data Set Formatter is the first lesson. This is the easiest of the macros to understand and utiliz.
- The Output Formatter is the second lesson. An exercise is included, and the output is uploaded to the server.
- The Tabulate Formatter, is the third lesson and focuses on the many options that the formatting macros provide.

The fourth section is an introduction to ODS. This includes a lesson in the ODS HTML destination. An exercise is provided in which a web page is developed and uploaded to the server automatically by using the FTP Access Method.

When the student has completed these five sections, the student should have three pages online, beginning with the home page created by the DATA _NULL_ step in the first interactive demo. This home page links to the pages created by the Output Formatter and ODS in the subsequent two exercises. A review of the material is presented in a final section.

FLOW, INTERACTIVITY AND FURTHER REVIEW

The website adheres to three principles that serve as useful guidelines for any instructional website: flow, interactivity, and further review.

FLOW

By its very nature a website allows students to take any "path" they desire. They can take a short excursion into supplementary material, they can back-up and review material, they can forge ahead and skip material they already know. This is the beauty of the web and our pages are designed to facilitate this behavior. However, as with any instruction, the flow of topics sometimes needs to be linear. In our case, we wanted the students first to learn basic HTML and then how to upload files using FTP, before moving to producing web output with a SAS DATA _NULL_ step. We accomplished this simply by the sequence of the pages.

To facilitate this we employ three links. The first link takes the student to the next lesson in the linear fashion as described above. It is placed directly after the lesson and is clearly marked as "Next".

The second and third links are in the address bar at the bottom of the page. The second link takes the student to a site index that allows the student to jump to any page on the website if they choose to review a topic later. The third link takes the student to a "Frequently Asked Questions" page. The FAQ extends the interactivity between the instructor and the student by providing a forum outside of the classroom for students to ask questions and receive answers.

Just as the sequence of the pages facilitates flow, so does the use of hyperlinks whenever there is a need for an example. The student who has no need for an example can ignore the link and stay focused on the lesson. But for the students who need an example, or another reference, the link can make a significant difference in their understanding of the material. A tip when creating such links is to eliminate the need to navigate backwards through a maze of pages to return to the lesson. To do this, add the attribute *target="_noframes"* to the hyperlink tag as follows:

```
<a href="http://anotherwebsite.com/"
target="_noframes">
```

A new browser window will open when selected. If one has already been opened and has not been closed, the linked page will load in that window. Besides providing links to examples, we also found it was useful to provide links to data files as an alternative to having the students key in the data themselves.

INTERACTIVITY

A substantial body of research indicates that students learn more when they interact in the classroom as opposed to listening to a "talking head." The authors heeded this research and designed the website to maximize student interaction. For example, rather than just showing the student the appropriate SAS program for the job, the student has to write the SAS program, run it, and thereby generate the HTML output themselves. The trade-off is that this strategy takes more time, especially if some of the students are less experienced.

To help alleviate this problem, we use web scripting. For instance, to link to an example of code and the output it generates, we enhance the hyperlink, by referencing a Javascript function:

```
<A onClick="this.href='javascript:
openExample(\'splitscreen.html\')'" href..>
```

"openExample" is a Javascript function which generates a full-screen window (without toolbars or menu bars, etc.) when the link is clicked. This full-screen window consists of three frames: a menu from which several examples can be selected, and windows for the SAS code and output, which appear side-by-side. Selecting one item in the menu opens the SAS code and output windows. We found that by displaying the code and output simultaneously the student gets a better understanding of how the codes produces the output.

In keeping with our efforts to maximize interactivity, we provide another link on the menu that gives the student the option of downloading the code and running it on their own computer. The result is a significant time saving as the student need not download any code nor run the code to see the output.

Another opportunity we discovered for enhancing interactivity using web scripting techniques, involves the students acquiring server permission for web publishing. On the University's server students and employees are allowed an online directory called "public_html" for web publishing. They must register to receive the necessary permission before the pages become accessible online. At UNC, this process can take up to an hour.

Our instruction requires that the students use this directory for storing their static web applications. To ensure that students have the necessary permissions set -- before they need them – we use a very simple Javascript-enhanced HTML form at the beginning of the website:

```
<FORM ACTION="" NAME="myform" METHOD="GET"
NCTYPE="text/plain"
onSubmit="document.myform.action =
'http://www.unc.edu/~' +
document.myform.userid.value" + '/'>
<P>Your UserID:
<INPUT TYPE="TEXT" NAME="userid"> </P> </FORM>
```

This form allows students to check whether they already have permission, and if not, allows them the opportunity to obtain the permission. The form reappears on the site wherever such access is required.

FURTHER REVIEW

The website concept empowers the student by encouraging them to revisit the material for review and further development. The two-hour class allows only an overview of several concepts and tools. In a situation where copious note-taking is not worthwhile, this provides the student with an opportunity to pick up what they can in class with the reassurance that they can return to the material later.

CONTENT CONTROL AND PORTABILITY

A website must be planned and designed properly. First of all, content and function, though inseparable to the end-user, are different elements to the webmaster. On the authors' site, the content consists of (HTML) markup. Functionality, alternatively, is augmented by cascading style sheets and Javascript source files. The remainder of this section provides some insight into how we were able to implement the user elements described above by adhering to this principle.

Cascading style sheets are a web standard for providing uniformity of formatting to a website. All of the fonts, their sizes, and their appearance (bold, italics, etc.) can be controlled from one central location. For instance, the following tells the browser that any text following a paragraph tag (<P>) should be written, if available on the computer, using Helvetica, Arial or Genevas fonts, in a ten-point font size, and in black:

```
P {
font-family:Helvetica,Arial,Genevas;
font-size:10pt;
color:#000000;
}
```

Uniform formatting is pleasing to the eye, but the benefit goes beyond simple aesthetics. Providing a central location to control the font size means being able to make "on-the-fly" changes to the font size of nearly all of the site's content in seconds. Referring to the paragraph formatting above, it would take only a moment to change the font size from 10 to 16. This means it can be done during the class, if needed. If a cascading style sheet were not used, then it would be impossible to do this – there would be hundreds of font size references spread throughout the website.

The Javascript source files, of which there are two, serve two purposes: they provide a place for nearly all of the Javascript functions on the site which give the site its interactivity, and they provide a place for adding uniform content. Uniform content means that content can be controlled from a specific location, much in the way that uniform formatting is provided by CSS. If there is a header or a footer that is on every page of the website, a Javascript source file can provide it. In fact, on the instructional site, a header consisting of a title block, some attractive images, the date and some links are added, as is a footer consisting of a "last modified" date, e-mail addresses, and links.

Uniform content was traditionally provided by Common Gateway Interface (CGI) in the form of server side includes. There are many problems and issues to resolve in implementing server side includes. It requires access to CGI-BIN on the web server. Many enterprises and organizations, the University included, have a policy against issuing access to the CGI-BIN directory on the server for security reasons. Second, it requires knowledge of PERL or other CGI scripting. Third, it does not provide access to Javascript functionality which allows for some variability to the content (see below for more). Finally, in order for the include to work on a webpage, the webpage must be retrieved from a server; if the computer goes offline, it will not work.

The Javascript source files can be thought of as "client-side includes" which are more easily implemented and functional than server-side includes. Javascript source files allowed us to include functions within our uniform content to provide for some variability. Each page, for instance, has a uniform header block that contains an image that tells the user which page of the lesson they are on. The entire block, however, including this image – which changes on each page of the lesson – is included as uniform content in the Javascript source file. The reason the source file is able to write a different image to each page accordingly is because a hidden attribute called "IMAGE" is passed from the .html page to the source file and interpreted by a Javascript function called "getImage", which tells the source file which image to write to the browser. The hidden attribute is written to the .html page, along with two others, as follows:

```
<form name="HIDE">
<input type="hidden" name="IMAGE" value="overview">
<input type="hidden" name="NEXT" value="htmlnotes">
<input type="hidden" name="PREVIOUS" value="home">
</hide>
```

The "NEXT" and "PREVIOUS" attributes tell the browser which page is next in the lesson and which came before it. The "NEXT" attribute is written to the next page link at the bottom of each lesson page (using "getNext"), telling the browser what .html page should be linked, and telling the browser what text to write to denote the link ("getLink"). For instance, referring to the above form, the following function in the Javascript source file

```
function getNext()  {
return(document.HIDE.NEXT.value)
}
```

interprets the value "NEXT" from the form. Then it is written to the .html file using

```
document.write('<a href="' + getNext() + '.html">');
getLink()
document.write('</a>');
```

(Note the HTML inside of the Javascript document.write methods.) The getLink function determines what goes between the and :

```
if(document.HIDE.NEXT.value == 'htmlnotes')
{document.write('Next:  A Brief Introduction to HTML')}
```

The benefit of using the getLink function, is that if there is no reference to a value, then it doesn't write anything to the page. When there is no next, sequential, page this is preferred.

The other use for "NEXT", and the reason for including "PREVIOUS", is a keystrokes script that provides some protection against distractions and provides both the student and instructor with a simple method of navigation. Instead of using the mouse to locate and select hyperlinks on the lesson pages, keystrokes have been programmed to move forwards and backwards through the pages. If the instructor or the student presses the "f" key, the Javascript "KeyPress" interprets that to be a link to the next page. Similarly, if "b" is pressed, then the function interprets it to be the value of "PREVIOUS".

COMPUTER BASED TRAINING

Although our website was not meant to compete with CMC (Computer Mediated Communication) nor any other asynchronous or dispersed means of computer-based instruction, we do share many of the same goals. One such goal is the delivery of a platform-independent means of instruction. The website is portable and non-proprietary, as is true with most Internet applications. There is modest platform dependence in the sense that some applications perform well on certain browsers and poorly on others.

Another goal that website instruction shares with other methods of computer-based instruction, is the opportunity for self-instruction. Self-instruction when learning how to create static web pages seems especially appropriate since the pages themselves serve as examples of the desired results.

CONCLUSION

This presentation covers many of the advantages of using website

instruction. Although our subjects, the use of the SAS Web Formatting Tools and ODS, are particularly amenable to website instruction, we believe other topics are equally appropriate for website instruction.

The important thing is that the website incorporate the principles of flow, interactivity, and review. If these guidelines are followed and the web pages present a logical progression of ideas and materials, the website can provide a powerful tool for instruction.

CONTACT INFORMATION

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

Roderick A. Rose
Jordan Institute for Families
CB 3550
301 Pittsboro St.
Chapel Hill, NC 27599
Email: rarose@email.unc.edu

Sally S. Muller
Jordan Institute for Families
CB 3550
301 Pittsboro St.
Chapel Hill, NC 27599
Email: sally@email.unc.edu
Work Phone: 919-843-7798