Paper 017-2013

The SAS® Output Delivery System: Boldly Take Your Web Pages Where They Have Never Gone Before!

Chevell Parker, SAS Institute Inc.

ABSTRACT

HTML output is one of the most effective and portable methods of disseminating information within an organization. Using a variety of techniques in the SAS® Output Delivery System (ODS), you can create HTML output that increases the visibility and functionality of your web pages. This paper discusses how to use these ODS techniques to deliver web content specifically for mobile devices, web content for both mobile and desktop devices, and web content specifically for desktop devices. In the first two categories, the paper discusses the challenges and solutions for both types of web content. For desktop devices, the paper discusses how to extract data from web pages and place it into pivot tables for data-visualization purposes in Excel.

INTRODUCTION

Advancements in technology make this an exciting time in which to live. Many options are now available for developing spectacular web content for both desktop and mobile devices. Web content has become intertwined in every facet of our lives—from entertainment to shopping, from school to work, and everything in between. As technology continues to advance, users need better methods for delivering content that is easy to read and that is consistent across various types of devices. This paper discusses the use of the SAS Output Delivery System along with advancements in cascading style sheets (CSS), and HTML5 to help you develop content for both mobile and desktop devices. The discussion details three categories of web content:

- web content specifically for mobile devices
- web content for both mobile and desktop devices
- web content specifically for desktop devices

Each of these sections cover some of the issues that developers face in creating and rendering web content as well as solutions for providing readable and consistent content using ODS, CSS, and HTML5.

WEB CONTENT FOR MOBILE DEVICES

The mobile revolution is exciting because it is a technology that truly permeates our daily lives. Mobile technology's portability means that it can go everywhere that people go. Late last year, Nielson reported that 50% of all cell phone users now own smart phones. This report marks the first time that this statistic has crossed the 50% line for the U.S

Despite their flexibility and power, mobile applications offer some challenges that are unique to their genre when it comes to delivering content. This section examines some of those challenges and discusses some of the latest mobile technology that offers solutions for providing outstanding mobile--to-web output.

When you deliver output on mobile devices, there are quite a number of issues to consider that you do not have to consider for desktop computers. Those issues include the following:

- Generated output needs to be accessible from a desktop browser as well as from mobile devices.
- Pages need to render on all devices very efficiently with a consistent look and feel.
- Pages and images should load quickly on all mobile browsers.
- The viewport and screen sizes on mobile devices have limitations.
- The display format needs to account for phones and tablets that use dual orientation.
- Fonts that you are used to accessing on a desktop machine are not always available on mobile devices.

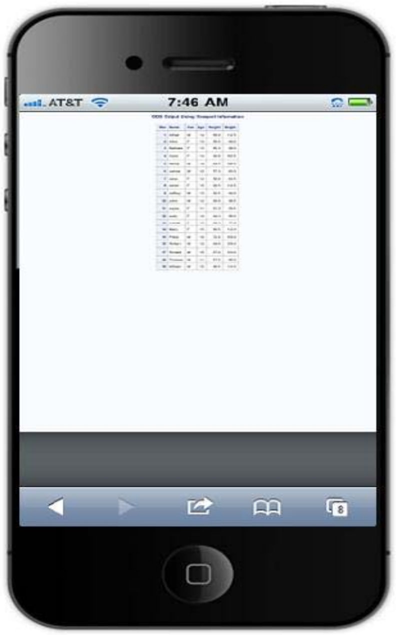
Despite many of the challenges presented here, the technology has come a long way in a very short time. In the past, mobile applications had to be generated with the native iOS or Android applications. Now, however, you can generate applications with SAS ODS Markup language, JavaScript, HTML, and CSS, all of which enable people to use existing skills, thereby reducing the learning curve. The following sections explain how to use these technologies to configure a mobile device's viewport for optimal size, how to generate consistent output across mobile devices, and how to create mobile applications. The sections also address some of the issues mentioned above.

CONFIGURING THE VIEWPORT FOR OPTIMAL SCREEN SIZE

On a mobile device, the *viewport* is the area where web content is available for viewing. When you view a website as a mobile page, the website assumes that you are viewing the page with a desktop machine. As a result, the rendering is set for a desktop machine rather than for the mobile device. That can cause readability issues because the width for mobile devices is so different from that for desktop devices. The default viewport size for the Apple iPhone is 980 pixels, and for android phones it is 800 pixels. The challenge, then, is to find the optimal resolution for displaying content as soon as it is loaded to a phone or tablet browser rather than having to enlarge the window to view the content. On a desktop computer, horizontal scroll bars are added to the browser when content is larger than the viewport. However, adding scroll bars is not an option for the mobile browser.

For example, consider the iPhone. This device is one of the newer phones with a dual display; that is, you can view it in either portrait or landscape orientation. The phone's visible area has a width of 320 pixels for the portrait orientation and 480 pixels for landscape orientation.

Now suppose that you have some output that is generated with the SAS ODS HTML destination. By default, the output appears to be zoomed out on the iPhone display, as shown here in **Display 1**.



Display 1. Unreadable Text on an iPhone

You can overcome this visibility problem by setting the viewport size with a viewport **`<meta>`** tag. There are parameters (such as height and width) that are available in this tag that help correct unreadable content on a mobile device. It is important to change the viewport width for web applications or pages that are designed for devices with smaller screens, especially those where content is narrower than 980 pixels. Such a modification requires only a single-line change to the HTML file. You simply add a viewport **`<meta>`** tag to the HTML file. The basic tag looks similar to the following:

```
   <meta name="viewport" content="width=device-width">
```

The *viewport* **`<meta>`** *tag* gives you control over the page characteristics such as height and width, the scale , dots per inch (DPI) density, the zoom level, and so on. For example, suppose the content in Display 1 has a width of 320 pixels (px). You can set the width to 320px and the page displays nicely. Because the device width of the screen is 320px, you can also set the height, making the output the optimal size. However, hardcoding this value can be a problem. Hardcoded values might not be optimal for another device such as the iPad. Therefore, the recommended method for solving this problem is to use two constants (**`device-width`** and **`device-height`**) in the **`<meta>`** tag that automatically adjusts the height and width that are appropriate for your device. Changing the height and width in this manner is the most important change for mobile websites. The two constants enable you to match the content to the device width, enabling the content to fit correctly regardless of the device's orientation.

You can create the necessary META tag by using the ODS HTML destination with the METATEXT= attribute. The following sample code creates a meta tag that uses the **`device-width`** constant to make the content in the display more readable. The META tag uses the CONTENT= attribute to incorporate the **`device-width`** constant, while the USERSCALABLE= attribute specifies, in this case, that users cannot zoom the window.

```
   ods html file="temp.html" style=htmlblue
            metatext='name="viewport"
            content="width=device-width"
            userscalable=no';
```

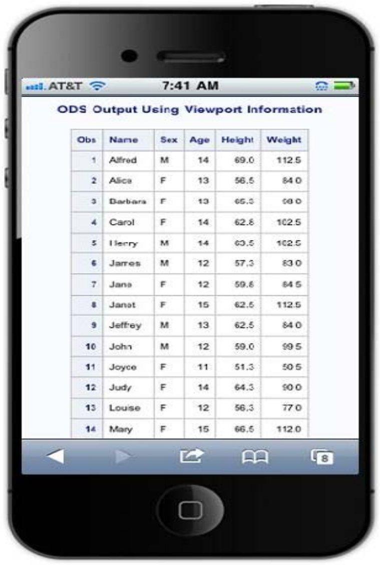
(code continued)

```
proc print data=sashelp.class;
run;

ods html close;
```

The ODS HTML destination in the sample code below generates the following **`<meta>`** tag in the HTML file:

```
<meta name="viewport"
      content="width=device-width userscalable=no">
```

Display 2 shows the more optimally sized content that is a result of the previous code.



AT&T 7:41 AM

ODS Output Using Viewport Information

Obs	Name	Sex	Age	Height	Weight
1	Alfred	M	14	69.0	112.5
2	Alice	F	13	56.5	84.0
3	Barbara	F	13	65.3	98.0
4	Carol	F	14	62.8	102.5
5	Henry	M	14	63.5	102.5
6	James	M	12	57.3	83.0
7	Jane	F	12	59.8	84.5
8	Janet	F	15	62.5	112.5
9	Jeffrey	M	13	62.5	84.0
10	John	M	12	59.0	99.5
11	Joyce	F	11	51.3	50.5
12	Judy	F	14	64.3	90.0
13	Louise	F	12	56.3	77.0
14	Mary	F	15	66.5	112.0

Display 2. Optimally Sized Content

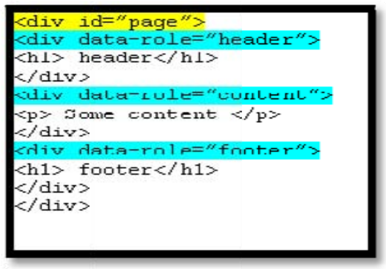
GENERATING WEB CONTENT THAT IS COMPATIBLE ACROSS MOBILE DEVICES

jQuery Mobile is a powerful web framework (or, a JavaScript library) that enables you to make your applications and web pages accessible to mobile devices, smartphones, tablets, and desktop devices. This JavaScript library is entirely free, and you can download it or use a hosted version. The jQuery Mobile framework is a unified HTML5-based interface for mobile platforms. This framework is all built on top of HTML5, which offers tremendous functionality. jQuery Mobile uses HTML5 **`data-*`** attributes to allow for markup-based modification of output. With the framework, you can change data roles or layouts by modifying **`<div>`** tags, add toolbars, generate transitions, add customized icons , display output using styles and themes, create custom headers and footers, and more. jQuery Mobile also enables you to generate output that is optimized for the mobile user. So unlike output on a desktop machine, jQuery Mobile output is also touch-enabled.

Basic Markup Required for Output That Uses the jQuery Mobile Framework

With a basic knowledge of HTML, you can use jQuery Mobile to generate dynamic web pages that are compatible across devices and browsers. To use the library, you simply include it by adding the following elements into the header of the HTML file. **Note:** The following statements are similar to what you should include, but be aware that jQuery Mobile is updated regularly.

```
<link rel="stylesheet"
        href="http://code.jQuery.com/mobile/1.2
        .0/jQuery.mobile-1.2.0.css"/
<script src="http://code.jQuery.com/jQuery-
1.8.2.min.js"></script>
<script src="http://code.jQuery.com/mobile/
           /1.2.0/jQuery.mobile-1.2.0.min.js">
</script>
```



```
<div id="page">
<div data-role="header">
<h1> header</h1>
</div>
<div data-role="content">
<p> Some content </p>
</div>
<div data-role="footer">
<h1> footer</h1>
</div>
</div>
```

Output 1. HTML5 Code for Use with the jQuery Mobile Framework

Modifying markup with jQuery Mobile enables you to create awesome web pages simply by modifying the structure of the HTML to change the way that the web pages are presented. The jQuery Mobile framework expects a basic HTML5 file that begins with the HTML5 **`<!doctype>`** declaration. The file is subdivided by **`<div>`** tags for the various sections of the page. As shown in **Output 1**, a partial, basic HTML5 file contains individual **`<div>`** tags for page and content as well as tags for any optional headers and footers. This basic HTML layout enables you to take advantage of jQuery Mobile framework's accessibility features.

ODS Tagsets and the jQuery Mobile Framework

ODS Markup Language is the perfect tool to interface with the jQuery Mobile Framework in order to generate various layouts. You use ODS Markup Language to create a customized tagset that uses the jQuery Framework. For details about how and where to store your tagsets, see "Base SAS: ODS Markup Resources." (**support.sas.com/rnd/base/ods/odsmarkup/index.html**)

There is an available jQueryMobile tagset that enables you to modify layouts, themes, transitions, icons, toolbar additions, footers and headers, and so on. To download the jQueryMobile tagset, see "Base SAS: ODS Markup Resources." (**support.sas.com/rnd/base/ods/odsmarkup/index.html**)

The next sections discuss how to implement the JQueryMobile tagset to create the following web-page layouts: default, page, collapsible, and tab.

The Default Layout

If you use the JQueryMobile tagset without any options, the layout uses the default option DATA_ROLE=”DEFAULT”. This tagset uses the power of jQuery to optimize the presentation of the output across browsers. The output looks much the same as the output that is shown previously in **Display 2**. The default layout is better when the output is from a single output or SAS procedure

The Page Layout

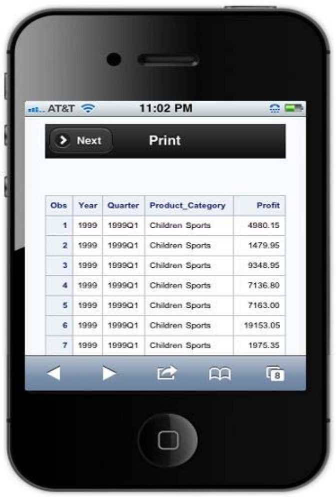
For web content where you use multiple procedures, you might want to use the page layout. The *page layout* offers alternate ways of viewing your output when you have multiple procedures. That is, *page layout* enables you to step through your output page by page. To change a web page to page layout, you use the DATA_ROLE=”PAGE” suboption in your ODS TAGSETS statement, as shown in the following example.

```
ods tagsets.jQueryMobile file="temp.html" options(data_role="page");

proc print data=sashelp.orsales(obs=50) nowd;
   columns var year quarter product_line product_category profit;
run;

proc report data=sashelp.class(obs=50) nowd;
run;

ods tagsets.jQueryMobile close;
```

With page layout, the HTML5 file that is created contains a **`<div>`** tag for each section of output, and each tag has a unique ID that is linked internally to individual pages. This configuration enables output to be displayed as concisely as possible.

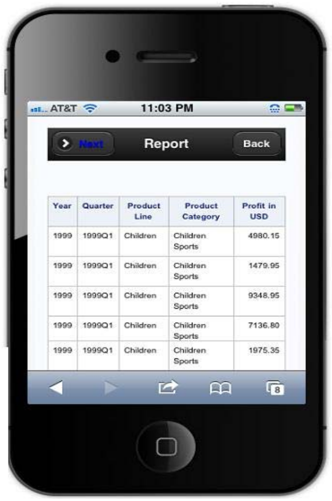
By default, the procedure name is added to the mobile device's toolbar, and you can modify the name with the HEADER= option. Other options are also available for use in this layout. For example, you can use the DATA_TRANSITION= option, which specifies the transition method that is to be used when you navigate between the reports.

As shown in **Display 3** and **Display 4**, the page layout offers a page-by-page view of the output when there are multiple procedures. Because the pages do not have a scroll bars, this type of output enables you to click a button on the toolbar to select each procedure output. From the second page and onward, a **`Back`** button is added automatically so that you can navigate back to the previous output.



Obs	Year	Quarter	Product_Category	Profit
1	1999	1999Q1	Children Sports	4980.15
2	1999	1999Q1	Children Sports	1479.95
3	1999	1999Q1	Children Sports	9348.95
4	1999	1999Q1	Children Sports	7136.80
5	1999	1999Q1	Children Sports	7163.00
6	1999	1999Q1	Children Sports	19153.05
7	1999	1999Q1	Children Sports	1975.35

Display 3. Web Report Created with ODS and the jQueryMobile Tagset



Year	Quarter	Product Line	Product Category	Profit in USD
1999	1999Q1	Children	Children Sports	4980.15
1999	1999Q1	Children	Children Sports	1479.95
1999	1999Q1	Children	Children Sports	9348.95
1999	1999Q1	Children	Children Sports	7136.80
1999	1999Q1	Children	Children Sports	1975.35

Display 4. Second Page of the Report with the `Back` Button Added

The Collapsible Layout

Displaying a lot of output at once in a desktop browser is not problematic, but it is a problem for mobile browser where space is limited. In such cases, you can use a *collapsible layout*. This type of layout adds collapsible output for procedures on the same page. The output for each procedure is labeled, and the information is collapsed, by default. But this layout enables you to expand and view only the information you want to see. The SAS procedure name is used as the headings unless you change them by using the ODS PROCLABEL statement.

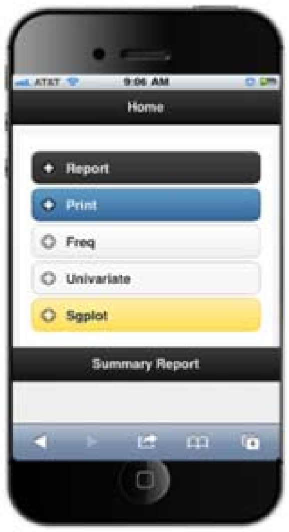
To generate the collapsible output, you use the tagset option DATA_ROLE="COLLAPSIBLE", as shown in the following code:

```
ods tagsets.jQueryMobile file="temp.html" options(data_role="collapsible"
                                                  data_theme="b"
                                                  footer="Summary Report";
ods tagsets.jQueryMobile options(data_content_theme="a");

proc report . . .options. . .;
proc print . . .options. . .;
proc freq . . .options. . .;
proc univariate . . .options. . .;

ods tagsets.jQueryMobile close;
```

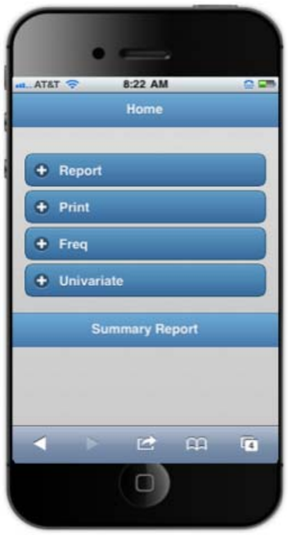
This code creates collapsible pages, and it also incorporates themes. As mentioned previously, the code uses the DATA_ROLE="COLLAPSIBLE" option to create the collapsible list. The sample program also includes a few other options: DATA_CONTENT_THEME=, the FOOTER=, and the DATA_THEME= options. (A HEADER= option is also available, but it is not used in the sample program.)

With the DATA_CONTENT_THEME= option, you can modify the style for each of the headers used in the collapsible list. The DATA_THEME= option can be added to all of the layouts, and you have five theme options available: A, B, C, D, and E. You can also create new themes. These themes affect items such as the toolbar colors and the background colors. For details about themes A-E and creating new themes, see the jQuery "Themes" documentation. (`jQuerymobile.com/demos/1.2.0/#/demos/1.2.0/docs/api/themes.html`)

Display 5 illustrates the use of the DATA_CONTENT_THEME= option for the list items, and **Display 6** illustrates the use of the DATA_THEME="B" option.



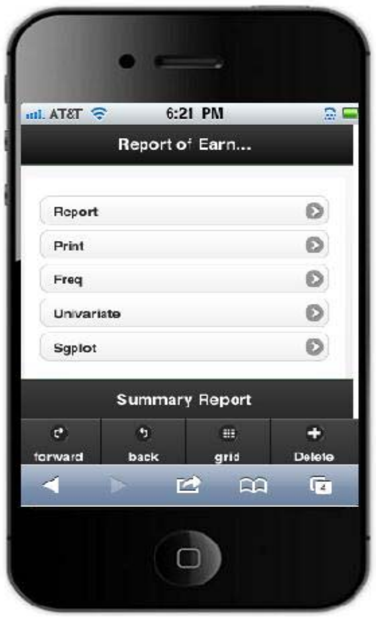
Display 5. Collapsible Page with Individual Data Content Themes



Display 6. Collapsible Page with an Overall Theme

The jQuery Mobile framework also gives you the ability to add standard icons to the bottom of the display. In addition, you can add personalized icons. as well as specify a location for the icons. You can apply other options as well. The following sample code uses the DATA_EXPANDED_ICON= and DATA_COLLAPSED_ICON= options to specify which icons to use when output is expanded or collapsed. The program also incorporates the DATA_ICONPOS= option, which specifies the location for the icon, and the DATA_MINI= option, which displays a compact and smaller view of the list.

```
ods tagsets.jQueryMobile file="temp.html"
options(data_role="collapsible"
      data_theme="c"
      header="Report of Earnings"
      footer="Summary Report"
      data_expanded_icon=”arrow-l”
      data_collapsed_icon=”arrow-r”
      data_iconpos=”right”
      data_mini="yes"
      footer_icon="forward,back,grid,delete);
. . .procedure statements. . .
ods tagsets.jQueryMobile close;
```



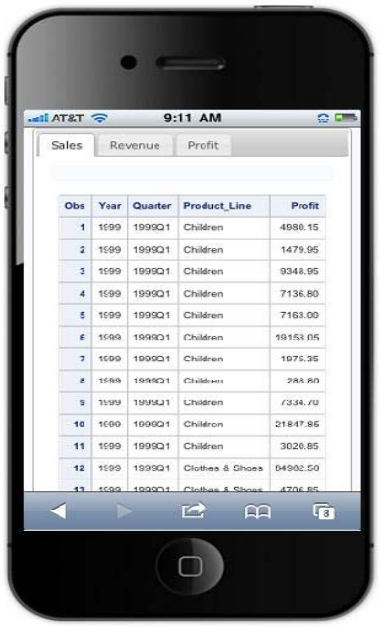
Display 7. Content with Icons Added

Display 7 shows the output that is created with the previous sample code.

The Tab Layout

The *tab layout* adds navigation tabs to your output. This layout enables you to use tabs to access individual output for multiple procedures from the same page. The following code sample uses the WEB_TABS= option to create tabs. This option specifies a name (**`Sales`**, **`Revenue`**, and **`Profit)`** for each tab, separated by commas, for each procedure of DATA step.

```
ods tagsets.jQueryMobile
   file="temp.html"
   options(web_tabs=”Sales,Revenue,Profit”);
proc print data=sashelp.prdsale;
run;

proc report data=sashelp.prdsale nowd;
run;

proc report data=sashelp.orsales;
run;

ods tagsets.jQueryMobile close;
```



Obs	Year	Quarter	Product_Line	Profit
1	1999	1999Q1	Children	4980.15
2	1999	1999Q1	Children	1479.95
3	1999	1999Q1	Children	9348.95
4	1999	1999Q1	Children	7136.80
5	1999	1999Q1	Children	7163.00
6	1999	1999Q1	Children	19153.05
7	1999	1999Q1	Children	1975.35
8	1999	1999Q1	Children	288.80
9	1999	1999Q1	Children	7334.70
10	1999	1999Q1	Children	21847.86
11	1999	1999Q1	Children	3020.85
12	1999	1999Q1	Clothes & Shoes	64902.50
13	1999	1999Q1	Clothes & Shoes	4706.85

Display 8. Web Content with Navigation Tabs

The tabs that are added, as shown in **Display 8**, enable you to navigate to any page of the output by touching the associated tab. As with previous layout examples, you can use the DATA_THEME= option and the FOOTER= option with the tab layout. **Note:** Headers (via the HEADER= option) are not used in the tab layout.

CREATING MOBILE APPLICATIONS

Using ODS, you can create applications that are designed specifically for mobile devices rather than for a desktop device. For example, you can create an application that calls native devices, generate maps, or generate a web application to pass your current location.

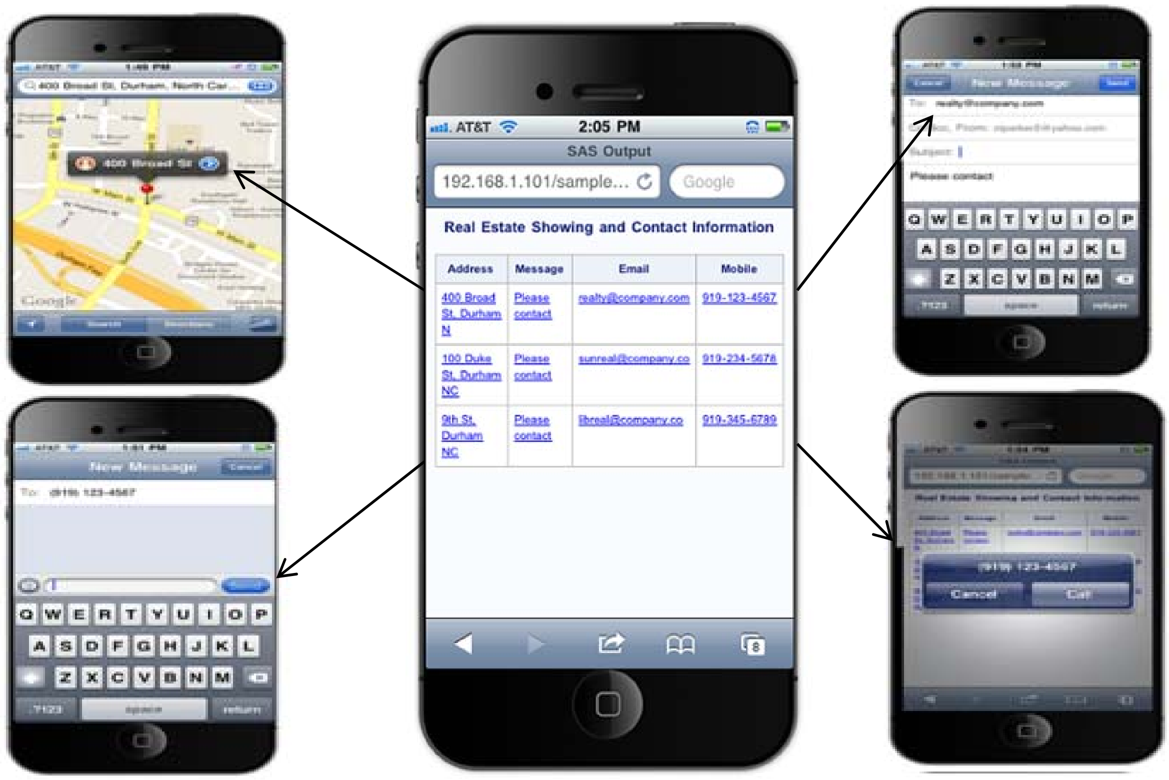
Calling Native Applications Using ODS

You can use various uniform resource identifiers (URIs) or protocols to call native applications for use on the iPhone. Some of the native applications that you can call include the click-to-call phone feature, e-mail, text messages, maps, video chat programs (Microsoft Skype and Apple FaceTime), and YouTube videos. For example, the following sample code uses the REPORT procedure with the appropriate protocol to create a link to activate the appropriate native application.

For this example, suppose that a real-estate agent pulls a list of showings for the day that contains addresses of the houses that are being shown, a message to the owners of the houses, the e-mail address of the agent, and the realty number. The following sample code links this information to various applications on the agent's iPhone.

```
data one;
   infile cards ;
   input Address $1-23 Message $24-38 Email $39-57 Mobile $59-72;
   cards;
400 Broad St, Durham NC Please contact realty@company.com  919-123-4567
100 Duke St, Durham NC  Please contact sunreal@company.com 919-234-5678
9th St, Durham NC       Please contact libreal@company.com 919-345-6789
;
run;

ods html path="C:\" file="mapapi.html" style=htmlblue
metatext='name="viewport" content="width=device-width userscalable=no"';

proc report data=one nowd;
   Title "Real Estate Showing and Contact Information";
   col address message email mobile;
   compute phone;
      call define('mobile',"url",cat("tel:",mobile));
      call define('message',"url",cat("sms:",mobile));
      call define('email',"url",cat("mailto:",email,"?body=",message));
      call define('address',"url",cat("http://maps.apple.com?Q=",address));
   endcomp;
run;

ods html close;
```

A COMPUTE block is added within PROC REPORT and links are created with the appropriate protocols to call the native applications (such as maps, texting, e-mail, and phone tasks), as shown in **Figure 1.**



Real Estate Showing and Contact Information

Address	Message	Email	Mobile
400 Broad St, Durham N	Please contact	realty@company.com	919-123-4567
100 Duke St, Durham NC	Please contact	sunreal@company.co	919-234-5678
9th St, Durham NC	Please contact	lbreal@company.co	919-345-6789

Figure 1. Using PROC Report to Generate Links That Call Native iOS Applications

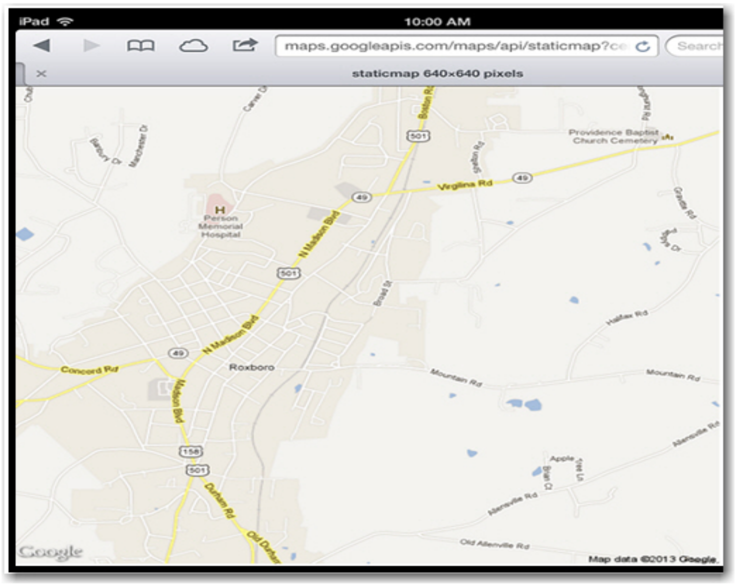
Integrating Maps into Mobile Devices

Maps are integrated into many of the major smart phones. You can integrate map applications easily into mobile devices by using one of the Google Maps APIs. The following code sample demonstrates the use of one such API.

```
ods html file="temp.html" style=htmlblue
         metatext='name="viewport"
         content="width=device-width
         userscalable=no”';

proc report data=one nowd;
   col address message email mobile;
   compute address;
      temp="http://maps.googleapis.com/maps/api/staticmap?center="||trim(left(a
      ddress))||'&zoom=14&size=400x400&sensor=false';
      call define(_col_,"url",temp);
   endcomp;
run;

ods html close;
```

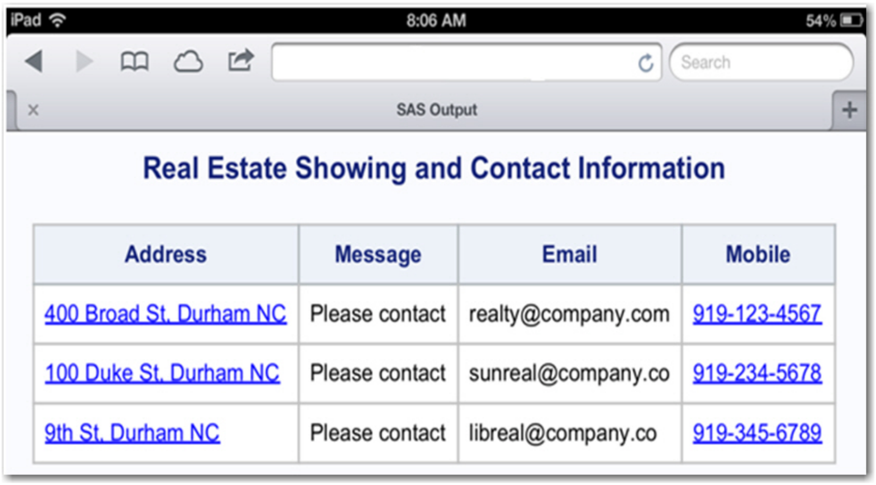
This code sample passes the value for ADDRESS to the Google Maps API. The API returns a static map of the location that is passed. For more information about the API, see the "Google Maps Image APIs: Static Maps API V2 Developer Guide." (**developers.google.com/maps/documentation/staticmaps/**)

When you pass a location such as a map address to the API, the API converts the location to latitude and longitude values and returns the map, as shown in **Display 9.**



Display 9. Map That Is Generated Based on an Address Passed to the Google Mapping API

PROC REPORT passes the address to the maps API, which generates a hyperlink in the table, as shown below. This application also works in desktop environments.



Real Estate Showing and Contact Information

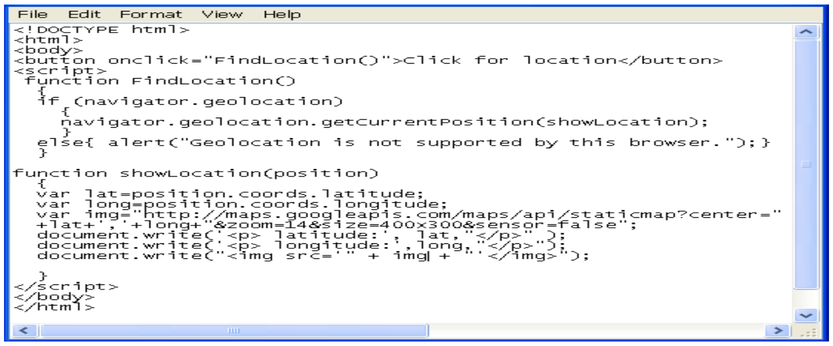
Address	Message	Email	Mobile
400 Broad St. Durham NC	Please contact	realty@company.com	919-123-4567
100 Duke St. Durham NC	Please contact	sunreal@company.co	919-234-5678
9th St. Durham NC	Please contact	libreal@company.co	919-345-6789

Display 10. Real Estate Table That Is Generated with PROC REPORT

Detecting and Mapping the Current Location

Beginning with HTML5, there is the new Geolocation API that enables you to determine your current location by returning your current longitude and latitude. Social media portals such as Facebook enable you to use this technology to track your current location and share your location with others. This type of application offers much in the way of business use, especially in terms of personalizing customer experience. For example, you might create an application that sends ads or coupons to customers based on their locations. Or you might send people a list of restaurants based on their locations.

The previous example created links to various applications for a real-estate agent. This next example displays the final output, which is created using the ODS Report Writing interface. The creates another useful web application for the real-estate agent—an application that always returns an iPhone's current location and that can be added to the iPhone's home screen. The application, created in HTML5, passes the current latitude and longitude information in the Geolocation API to the Google maps API to display a map of the current location. The HTML5 code is as follows:



```
<!DOCTYPE html>
<html>
<body>
<button onclick="FindLocation()">Click for location</button>
<script>
 function FindLocation()
   {
   if (navigator.geolocation)
     {
     navigator.geolocation.getCurrentPosition(showLocation);
     }
   else{ alert("Geolocation is not supported by this browser.");}
   }

function showLocation(position)
   {
   var lat=position.coords.latitude;
   var long=position.coords.longitude;
   var img="http://maps.googleapis.com/maps/api/staticmap?center="
   +lat+','+long+"&zoom=14&size=400x300&sensor=false";
   document.write('<p> latitude:', lat,"</p>" );
   document.write('<p> longitude:',long,"</p>");
   document.write("<img src='" + img + "'</img>");

   }
</script>
</body>
</html>
```

Output 2: HTML5 Code That Detects and Writes an iPhone's Current Location

In this HTML, the first step is to use the `FindLocation` function to verify that the Geolocation object exists. If it does exist, the HTML creates JavaScript variables LAT and LONG with values for the current latitude and longitude. These variables are passed to the Google Maps API and the values are displayed when you click the **`Click for location`** button. Then the HTML adds the application to your iPhone home screen so that you can locate it easily. There are a couple of meta tags that are required in order to save this map as a web application that you can use to remove the address bar and to provide an image for the web application.

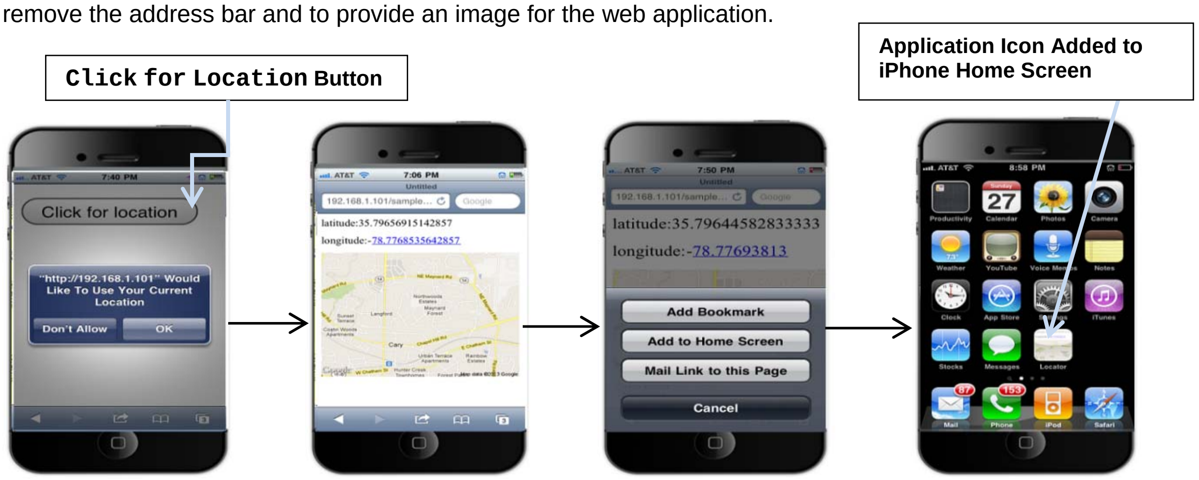

Click for Location Button
Application Icon Added to iPhone Home Screen
Click for location
"http://192.168.1.101" Would Like To Use Your Current Location
Don't Allow
OK
latitude:35.79656915142857
longitude:-78.7768535642857
latitude:35.79644582833333
longitude:-78.77693813
Add Bookmark
Add to Home Screen
Mail Link to this Page
Cancel
Locator

Figure 2. The Application Steps Created with HTML5 to Determine an iPhone's Current Location

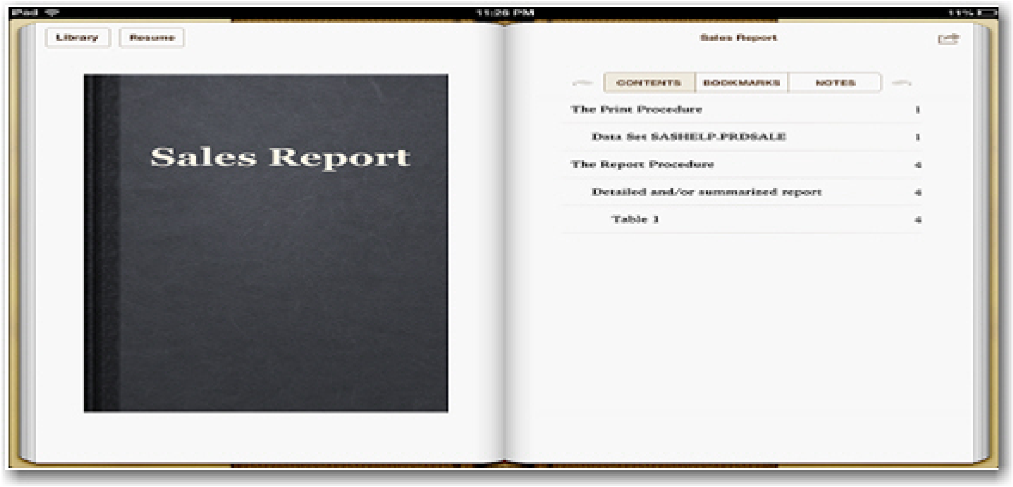
GENERATING MOBILE OUTPUT USING THE EPUB DESTINATION

Beginning in SAS 9.4, there is a new ODS destination (ODS EPUB) that generates files that are stored in an electronic book format. *EPUB*, short for **electronic publication**, is an open-file format for electronic books, newspapers, and magazines, and images. This format is readable by many e-book readers such as the Barnes and Noble Nook, the Amazon Kindle, and the Apple iPad and iPhone (using iBooks). With this format, you can create a cover page, add a table of contents, or store desired fonts within the file

The following sample code uses the ODS EPUB destination to generate an electronic-publication file:

```
ods epub file="temp.epub" title="Sales Report" newchapter=output
options(contents="yes");

proc print data=sashelp.prdsale(obs=50);
run;

proc report data=sashelp.orsales(obs=50) nowd;
run;

ods epub close;
```

The file that is generated is displayed below in the Apple iBooks application that is used by the iPhone and iPad for electronic publications.



Sales Report

The Print Procedure	1
Data Set SASHELP.PRDSALE	1
The Report Procedure	4
Detailed and/or summarized report	4
Table 1	4

Display 11. A Sample Sales Report That Is Generated with the ODS EPUB Destination

WEB CONTENT FOR MOBILE AND DESKTOP DEVICES

HTML5 is fast becoming the standard markup language for presenting web content on both mobile and desktop devices because most browsers support parts of both CSS3 (the latest CSS standard) and HTML5. This section discusses enhancements in both the CSS and HTML5 markup languages and what those enhancements mean in terms of web viewing on mobile and desktop devices.

ENHANCING OUTPUT USING CASCADING STYLE SHEETS

A *cascading style sheet* (CSS) defines the appearance and formatting of a document. That is, a *CSS* simply contains formatting instructions for elements such as size, color, or font location on the page. These style sheets enable you to make a distinction about the output that is rendered. For example, using CSS media properties, you can determine whether a style should be rendered on the screen or in print. These and other advanced CSS features are discussed in this section. The following topics explain some of the basics of CSS and how it is used within SAS as well as how to take advantage of new features in CSS3 from desktop and mobile devices.

Cascading Style Sheets: Overview

This section covers the various types of styles that (embedded, external, and in-line) are supported with CSS, the benefits of these styles, their structures, and how they are used. The section also briefly covers how to use ODS style attributes to interact with an existing CSS file, specifying style properties for various parts of output, and enhancements to the CSS3 standard.

CSS Types

There are three types of cascading style sheets: embedded, external, and in-line. You can use any combination of these style sheets in an HTML document. The *embedded* (or, internal) *style sheet is* created by default when you generate output with the ODS HTML destination. This internal style sheet is added directly in the header section (using the **`<head>`** tags) of the generated HTML file. The document that is created is very portable because the formatting instructions and the data reside in the same location.

The *external style sheet* is used in the **`<link>`** tag with the HREF= attribute, which points to the location of the CSS file that contains the style information. While an internal style sheet is portable, the external style sheet is the more powerful of the two. Because an external style sheet is stored as a separate file with a .css extension, you can use this file to modify the formatting for a single HTML file or an infinite amount of files. *External style sheets* also enable you to reduce the size of your HTML file by adding the style definition externally to the file. You can have a different style sheet for each individual device that you want to format. But because so many different devices are used to view web pages, it is not efficient to use individual style sheets. Whether you view a web page with a smart phone or a tablet or desktop browser, you want users to experience the same content across devices but in a native format that is designed specifically for each device type. With external style sheets, you can create one style sheet for phones, another style sheet for tablets, and another style sheet for the desktop browsers.

The *inline style* sheet is one in which style attributes are added directly in HTML tags in a document. They are often found in the **`<span> or <div>`** tags.

CSS Style Selectors

Cascading style sheets comprise selectors, style properties, and values. *Style selectors* are mechanisms that web pages use to associate various elements and tags to the styles. There are four basic of selector types: class selector, type selector, ID selector, and compound selector. The *class selector* consists of a class name that is preceded by a dot. The *type selector* is specified by one of the defined HTML elements or tags. The *ID selector* consists of a hash tag (#) prior to the ID name. Finally, the *compound selector* uses multiple selectors or a combination of selectors for a style property.

The following example demonstrates how some of the selectors are used along with a style property and a value in a common CSS declaration. This combination of a selector, a style property, and a value is known as a *CSS rule*. This particular CSS declaration specifies the **`.body`** class selector, the **`h1`** type selector, and the **`#item`** ID selector.

```
.body {color: red; background-color: #808080; font-size: 12px;
       font-style: italic; font-weight:bold}
h1 { color:red;  font-style:italic;}
#item {position:relative; color:red; font-size:10px;}
```

In addition to creating your own CSS files, you can create a CSS file based template style definitions that are shipped with SAS. For example, you can specify the standard template style by using the STYLE= option and the STYLESHEET= option in the ODS HTML statement, as shown below:

```
ods html stylesheet='c:\analysis.css'  style=styles.analysis;
```

To incorporate any existing CSS file, you can use the option STYLESHEET=(URL="*filename*.css"). Creating a .CSS file based on an existing style gives you a place to start if you already have a style that you like but want to tweak.

CSS Style Attributes

Procedures such as REPORT, TABULATE and PRINT that use the STYLE= option with the TEMPLATE procedure use various style attributes that interact with a style sheet. The style attributes mentioned in this section either use style information from an existing style or they enable you to write style sheet properties and values to parts of the output. For example, you can use the HTMLCLASS= attribute to associate a class name to various parts of the output either within the procedure or within a template style definition. The HTMLID= attribute enables you to associate an ID from a CSS style to various parts of the output from the procedure or template style definition. The HTMLSTYLE= attribute enables you to pass a CSS style property from a procedure or style definition.

CSSSTYLE= Option

Beginning in SAS 9.2, ODS includes the CSSSTYLE= option as its newest method of using cascading style sheets as a basis for styles. The CSSSTYLE= option makes use of the CSS engine. This combination of the CSSSTYLE= option and the engine enables you to use CSS files with all of the ODS destinations (PDF, TTF, HTML, and so on) that allow formatting. For more information on the CSSSTYLE= option, see the "References" section.

CSS3 Style Enhancements

The latest standard for CSS is CSS3. Newer smart phones now support much of CSS3, as do most of the popular browsers (for example, Microsoft Internet Explorer, Google Chrome, Mozilla Firefox, and Opera Software's Opera). Some of the technologies (such as Adobe Flash Player or scripting) that were only available in third-party products are now available or are soon-to-be available via CSS and HTML5. CSS3 contains many new features that were only available previously in a desktop publishing package. Those features include the following abilities:

- generating animations, gradients, font shadows, rounded borders and so on.
- rotating text. This ability has not been possible before with the exception of the Microsoft Excel special transforms that enable rotation of output either vertically or horizontally. Now, you can specify the degree to which you want to rotate text similar to the functionality that Excel provides.
- using media queries that enable you to detect a particular device. Within a media query, you can communicate with a device and retrieve useful information (for example, the width of the device). Then you can apply styles based on that information. There are several ways to specify media queries, depending on the device.

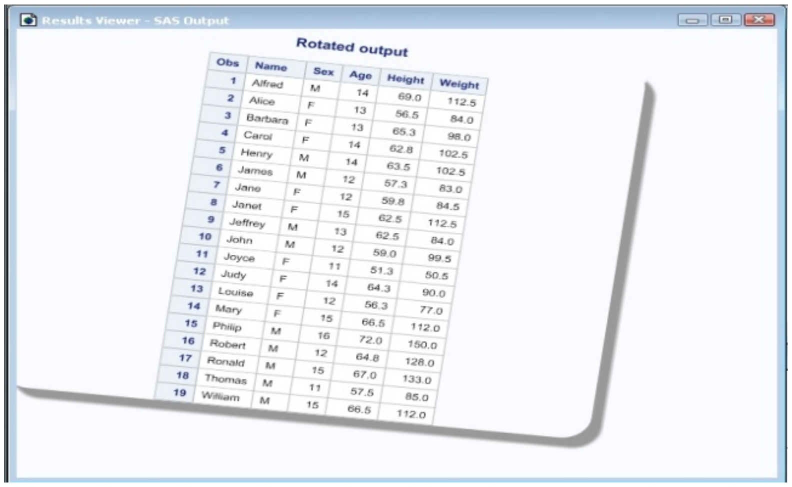
The following sections illustrate examples of using a media query and rotating text.

Rotating a Table and Including Rounded Corners and a Box Shadow

This first example uses the CSS3 **`rotate`** property to rotate the table and the titles in the SAS Results Viewer output. The example also uses CSS3 features to add rounded corners and a box shadow.

```
proc template;
   define style styles.test;
      parent=styles.htmlblue;
      class body / prehtml="<div style=""-ms-transform:rotate(7deg);
                                         border:2px solid;
                                         box-shadow: 10px 10px 5px #888888"">"
                   posthtml="</div>";
   end;
run;

ods html5 file="temp.html" style=styles.test;
proc print data=sashelp.class;
run;
ods html5 close;
```

This sample code generates the following output:



Rotated output

Obs	Name	Sex	Age	Height	Weight
1	Alfred	M	14	69.0	112.5
2	Alice	F	13	56.5	84.0
3	Barbara	F	13	65.3	98.0
4	Carol	F	14	62.8	102.5
5	Henry	M	14	63.5	102.5
6	James	M	12	57.3	83.0
7	Jane	F	12	59.8	84.5
8	Janet	F	15	62.5	112.5
9	Jeffrey	M	13	62.5	84.0
10	John	M	12	59.0	99.5
11	Joyce	F	11	51.3	50.5
12	Judy	F	14	64.3	90.0
13	Louise	F	12	56.3	77.0
14	Mary	F	15	66.5	112.0
15	Philip	M	16	72.0	150.0
16	Robert	M	12	64.8	128.0
17	Ronald	M	15	67.0	133.0
18	Thomas	M	11	57.5	85.0
19	William	M	15	66.5	112.0

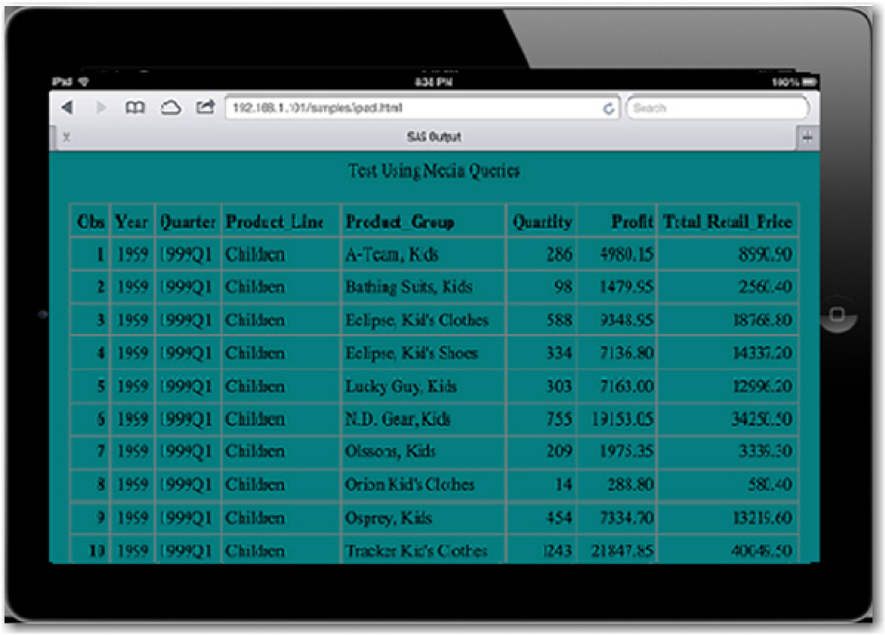
Display 12. Rotated Output with Rounded Corners and a Box Shadow Effect

Using a Media Query to Detect a Device Based on iPad Width and Orientation

The following example uses a media query that contains the widths of an iPad device for both portrait and landscape mode, and it further queries the device's orientation (landscape) to add a style for the iPad when only when the orientation is landscape. We could use this method to detect the device for any of the mobile devices that support the media queries.

```
   /* Adjust width for an iPad in landscape mode and change */
   /* background color.                                     */
@media only screen
and (min-device-width : 768px)
and (max-device-width : 1024px)
and (orientation : landscape) {
{body:background-color:green}
}
```

With this special break point (768px x 1024px), you can detect whether the orientation of the output is landscape or portrait, and you can apply a style based on that information.. For example, the style might be anything from setting the width or changing background color to more complicated style features. The sample code shown below instructs the iPad background color (which is normally white) to change to green when the iPad is tilted and the orientation is landscape.



Test Using Media Queries

Obs	Year	Quarter	Product_Line	Product_Group	Quantity	Profit	Total_Retail_Price
1	1959	1999Q1	Children	A-Team, Kids	286	4980.15	8590.50
2	1959	1999Q1	Children	Bathing Suits, Kids	98	1479.55	2560.40
3	1959	1999Q1	Children	Eclipse, Kid's Clothes	588	9348.55	18768.80
4	1959	1999Q1	Children	Eclipse, Kid's Shoes	334	7136.80	14337.20
5	1959	1999Q1	Children	Lucky Guy, Kids	303	7163.00	12996.20
6	1959	1999Q1	Children	N.D. Gear, Kids	755	19153.05	34250.50
7	1959	1999Q1	Children	Olssons, Kids	209	1975.35	3338.30
8	1959	1999Q1	Children	Orion Kid's Clothes	14	288.80	580.40
9	1959	1999Q1	Children	Osprey, Kids	454	7334.70	13215.60
10	1959	1999Q1	Children	Tracker Kid's Clothes	1243	21847.85	40048.50

Display 13. iPad Background Color Changed Using a CSS3 Media Query

ENHANCING THE DESIGN AND COMPONENTS OF YOUR WEB PAGES

As mentioned earlier, generating web pages has always been a favorite method in business organizations for disseminating information throughout those organizations. Given the popularity of this method, recent advancements in HTML make this an exciting time for incorporating Web pages even more into daily business activities. This section highlights some of the recent advancements of HTML and how you can take advantage of using these advancements with ODS to enhance your web pages generated.

New and Notable Features in HTML5

HTML5 is still a work in progress at this point, but most of the major browsers support many of the new HTML5 APIs and elements. The new HTML5 standard is a cooperative effort between the World Wide Web Consortium and the Web Hypertext Application Technology working group. Despite the fact that HTML5 is not yet complete, it is becoming the new standard for HTML and will revolutionize the way that web pages are used. HTML5 was created to overcome limitations in HTML4 and in the previous versions of HTML and XHTML. These earlier versions of HTML require the use of other third-party applications such as Adobe Flash Player and JavaScript to implement various types of functionality).However, many of these features are now standard items are now in the new HTML5 standard. This standard has been pushed by companies such as Google and Apple because it gives them capabilities they need for their mobile platforms.

Some of the new rules adapted for HTML5 include the following:

- New features should be based on HTML, CSS, DOM, and JavaScript.
- New elements (such as the **`<section>`**, **`<article>`**, **`<header>`,** and **`<footer>`** tags) have been added in order to better organize pages. HTML5 also uses a new **`<canvas>`** element that enables you to draw graphics as you go.
- HTML5 includes new form controls such as the calendar picker and date pickers.
- More markup is available to replace scripting.
- HTML5 is device independent.
- HTML5 reduces the need for third-party tools and JavaScript.
- HTML5 supports embedded scalable vector graphics (SVG) files.
- HTML5 supports the Geolocation API.
- HTML5 incorporates drag-and-drop capability.

More APIs are included for both audio and video (using the new **`<audio>`** and **`<video>`** elements). The standard also supports off-line content, which enables you to save information from content locally. As mentioned previously, the new **`<canvas>`** element, along with JavaScript or other scripting, .enables you to draw graphics or text as you go.

Support for embedded SVG files is a key advancement in HTML5. Vector graphics are a great way to deliver stunning visual results using minimal bandwidth, which is a bonus for mobile devices. SVG files are resolution independent, meaning that you do not need to think about how many pixels you have on your device. The resulting images always scale and are optimized by the browser to look great. You can also use GZIP to compress your SVG files However, not all browsers support this compression. When you use GZIP to compress SVG, the file extension is .svgz.

The features discussed in this section are just some of the many new features that are available in HTML5. SAS 9.4 ODS also contains many enhancements for HTML. The next section discusses those features and how they are implemented with the HTML destinations of ODS.

ODS Enhancements for HTML in SAS® 9.4

Starting with SAS 9.4, ODS contains the newest member of the markup destinations: the HTML5 destination. If you are using HTML5 technology, the new ODS HTML5 destination enables you to generate output based on the HTML5 standard. The default ODS HTML destination in SAS 9.4 is still HTML4, but that default destination is expected to change at some point in the future.

To invoke the new HTML5 destination, you specify the ODS HTML5 statement. If you want HTML5 to be your default form of HTML, you can modify the SAS registry using either the SAS Registry editor or the REGISTRY procedure to change the registry keys from HTML4 to HTML5.

If you look at the HTML file that is generated with this new ODS HTML5 destination, you will notice that the structure of the file is a little different than previous HTML files. The first tag that you see is the **`<!Doctype HTML>`** tag. This is the first tag that is required to create an HTML5 f-page.

HTML5 also contains elements that make the HTML source code more readable. In addition, the default styles for the HTML markup languages have changed, so the output looks quite a bit different. For HTML5, SVG is the default file format for the graphics files. One benefit of an SVG file is that it does not lose any quality when you zoom in or out, and SVG output can be printed at any resolution without losing any quality. The SVG format is an XML format that enables you to open the file in a text editor and modify the output. You can also embed SVG graphics in an HTML file that is created within SAS 9.4, which allows for greater portability of the file. Because the graphics are embedded, you no longer to e-mail the HTML file along with the supporting image files.. Internet Explorer9 and the majority of the major browsers handle this feature, but not all browsers can take advantage of it.

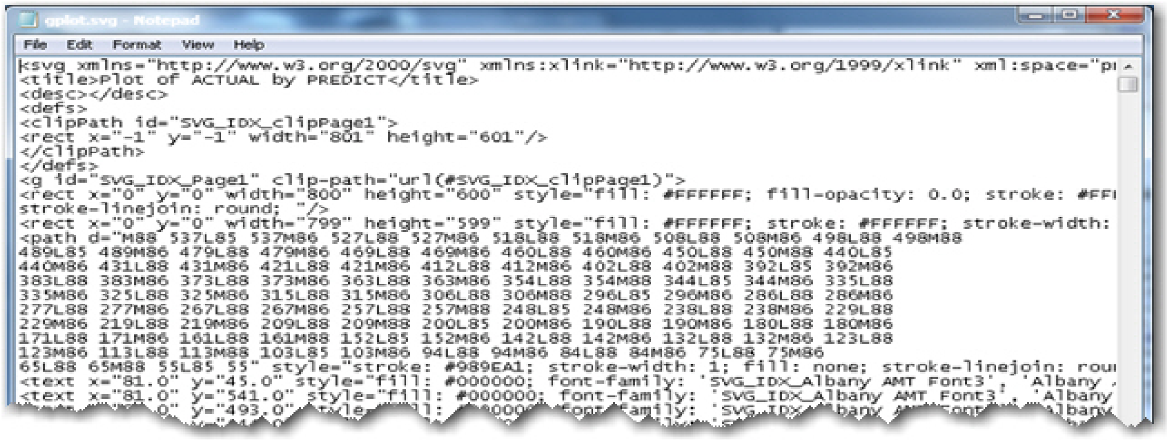
The following example demonstrates the use of the ODS HTML5 destination along with the SVG_MODE= option to generate an embedded SVG file with SAS 9.4.

```
goptions reset=all device=svg;
ods html5 options(svg_mode='embed');
title "Actual sales versus predicted”;
```

(code continued)

```
proc gplot data=sashelp.prdsale;
   plot actual*predict;
run;
quit;

ods html5 close;
```

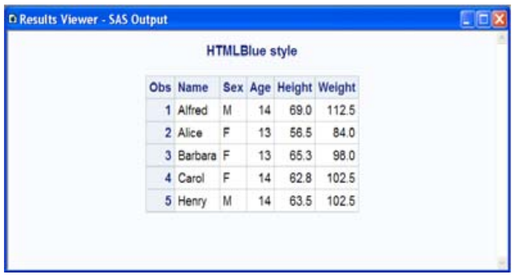
This code produces the following output, which shows the SVG file embedded in an XML file:



```
<svg xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink" xml:space="p
<title>Plot of ACTUAL by PREDICT</title>
<desc></desc>
<defs>
<clipPath id="SVG_IDX_clipPage1">
<rect x="-1" y="-1" width="801" height="601"/>
</clipPath>
</defs>
<g id="SVG_IDX_Page1" clip-path="url(#SVG_IDX_clipPage1)">
<rect x="0" y="0" width="800" height="600" style="fill: #FFFFFF; fill-opacity: 0.0; stroke: #FF
stroke-linejoin: round; "/>
<rect x="0" y="0" width="799" height="599" style="fill: #FFFFFF; stroke: #FFFFFF; stroke-width:
<path d="M88 537L85 537M86 527L88 527M86 518L88 518M86 508L88 508M86 498L88 498M88
489L85 489M86 479L88 479M86 469L88 469M86 460L88 460M86 450L88 450M88 440L85
440M86 431L88 431M86 421L88 421M86 412L88 412M86 402L88 402M88 392L85 392M86
383L88 383M86 373L88 373M86 363L88 363M86 354L88 354M88 344L85 344M86 335L88
335M86 325L88 325M86 315L88 315M86 306L88 306M88 296L85 296M86 286L88 286M86
277L88 277M86 267L88 267M86 257L88 257M88 248L85 248M86 238L88 238M86 229L88
229M86 219L88 219M86 209L88 209M88 200L85 200M86 190L88 190M86 180L88 180M86
171L88 171M86 161L88 161M88 152L85 152M86 142L88 142M86 132L88 132M86 123L88
123M86 113L88 113M88 103L85 103M86 94L88 94M86 84L88 84M86 75L88 75M86
65L88 65M88 55L85 55" style="stroke: #989EA1; stroke-width: 1; fill: none; stroke-linejoin: rou
<text x="81.0" y="45.0" style="fill: #000000; font-family: 'SVG_IDX_Albany AMT Font3', 'Albany
<text x="81.0" y="541.0" style="fill: #000000; font-family: 'SVG_IDX_Albany AMT Font3', 'Albany
```

Output 3. The Embedded SVG File That is XML Format

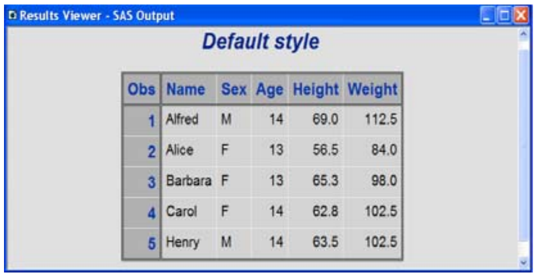
The SAS 9.4 HTML destinations and markup include some distinct changes to the default style that is used by the HTML and HTML5 destinations. In releases earlier than SAS 9.4, the default style that is named Default is used by the HTML destinations. Beginning with SAS 9.4, the HTML destinations use the HTMLBLUE style. (**Display 14**). This style is not new to users that are running SAS 9.3 because it is the default style for the SAS Display Manager output.



HTMLBlue style

Obs	Name	Sex	Age	Height	Weight
1	Alfred	M	14	69.0	112.5
2	Alice	F	13	56.5	84.0
3	Barbara	F	13	65.3	98.0
4	Carol	F	14	62.8	102.5
5	Henry	M	14	63.5	102.5

Display 14. Output Using the HTMLBLUE Style



Default style

Obs	Name	Sex	Age	Height	Weight
1	Alfred	M	14	69.0	112.5
2	Alice	F	13	56.5	84.0
3	Barbara	F	13	65.3	98.0
4	Carol	F	14	62.8	102.5
5	Henry	M	14	63.5	102.5

Display 15. Output Using the Older Default Style

WEB CONTENT FOR DESKTOP DEVICES

The ability to generate pivot tables is one of the most powerful features in Microsoft Excel. Pivot tables enable you to quickly summarize, analyze, and make sense out of large amounts of data. These tables are used by many decision makers because of the ease of use and effectiveness. Fundamental business questions are answered by dragging and dropping columns to different rows, columns, or summary zones within the pivot table. You can generate pivot tables using the ODS tableEditor tagset. The following sections explain how add data-visualization features (such as data bars and icons) to generated pivot tables. The discussion also explains how to create calculated columns.

EXPORTING PIVOT TABLES IN WEB PAGES

The *tableEditor tagset* enables you to export pivot tables from data on a web page to an Excel worksheet. To download the tableEditor tagset, see "Base SAS: ODS Markup Resources." (**support.sas.com/rnd/base/ods/odsmarkup/index.html**)

In Internet Explorer, you can use the tableEditor tagset to generate HTML web pages that contain pivot tables. To use a pivot table in data analysis, you export the table to Excel simply by clicking a button. The following sections discuss various ways you can use exported pivots tables in Excel. The first section explains how to add data visualization tools to table columns. The second section illustrates how to create calculated fields in your tables

Adding Data-Visualization Features to Table Columns

To add data-visualization features to table columns using the tagset, you need Microsoft Excel 2007 or later.. Beginning with Excel 2007, you have access to a number of data-visualizations tools that enable you to analyze numbers and deliver a visual representation of that analysis in your pivot table. This method is a great way to send summarized data if you do not want to add an additional graph.

This section discusses how to add visual data to table columns using icon sets, color gradients, and data bars within table cells.

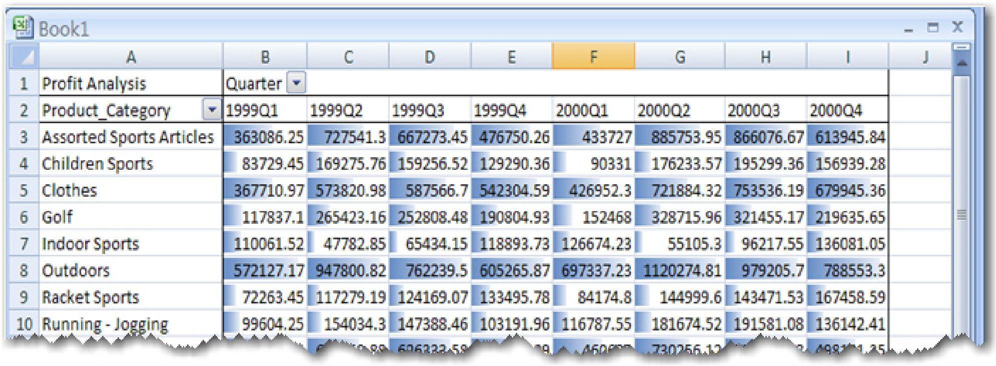
You can specify the tagset option FORMAT_CONDITION= and specify arguments such as DATABAR, ICONSETS, and COLORSCALE to add visual data to a table column**.**

Adding Data Bars

The following example uses the FORMAT_CONDITION= option with the DATABAR argument to add data bars to one or more columns in the table. Along with the DATABARS argument, you include the column number or range of columns that the data bar should include. This example prints output from the SASHELP.PRDSALE data set and adds data bars to columns 2-9.

```
ods tagsets.tableeditor file="temp.html"
options(format_condition="databar,2-9"
        pivotrow="product_category"
        pivotcol="quarter"
        pivotdata="profit"
        pivotdata_caption="Profit Analysis"
        pivot_grandtotal="no");
proc print data=sashelp.orsales;
           where quarter in("1999Q1","1999Q2","1999Q3","1999Q4",
                                "2000Q1","2000Q2","2000Q3","2000Q4");
run;
ods tagsets.tableeditor close;
```

The added data bars enable you to see quickly which products sold the best.



	A	B	C	D	E	F	G	H	I	J
1	Profit Analysis	Quarter								
2	Product_Category	1999Q1	1999Q2	1999Q3	1999Q4	2000Q1	2000Q2	2000Q3	2000Q4	
3	Assorted Sports Articles	363086.25	727541.3	667273.45	476750.26	433727	885753.95	866076.67	613945.84	
4	Children Sports	83729.45	169275.76	159256.52	129290.36	90331	176233.57	195299.36	156939.28	
5	Clothes	367710.97	573820.98	587566.7	542304.59	426952.3	721884.32	753536.19	679945.36	
6	Golf	117837.1	265423.16	252808.48	190804.93	152468	328715.96	321455.17	219635.65	
7	Indoor Sports	110061.52	47782.85	65434.15	118893.73	126674.23	55105.3	96217.55	136081.05	
8	Outdoors	572127.17	947800.82	762239.5	605265.87	697337.23	1120274.81	979205.7	788553.3	
9	Racket Sports	72263.45	117279.19	124169.07	133495.78	84174.8	144999.6	143471.53	167458.59	
10	Running - Jogging	99604.25	154034.3	147388.46	103191.96	116787.55	181674.52	191581.08	136142.41	

Display 16. Pivot-Table Output with Data Bars Added to Columns

Adding Icons

You can add meaningful icons pivot-table output in a similar fashion as the data bars. Using the FORMAT_CONDITION= option, you simply specify ICONSETS as the first argument. The second argument is the column number to which the icon should be applied. You can also add an optional, third parameter that specifies the type of icon (for example, arrows or flags) that is be added. Table 1 below shows the icons that you can add. Table 2 lists the types of icons you can specify.

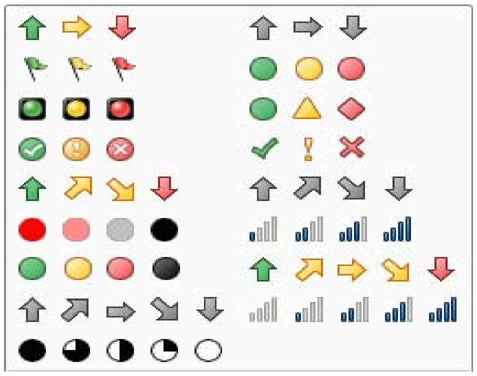


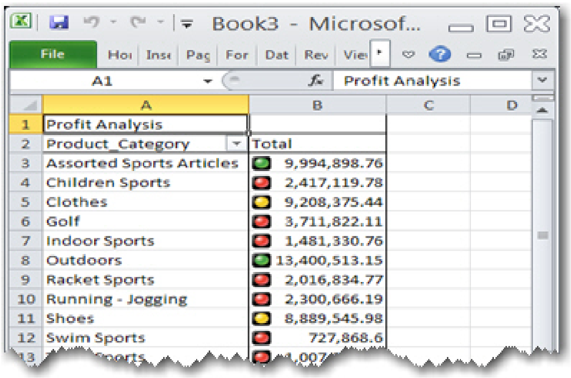
Table 1. Icon Sets

ICON VALUE	DESCRIPTION
1	Arrows 1
2	Arrows 2
3	Flags
4	Traffic Lights 1
5	Traffic Lights 2
6	Signs
7	Symbols
8	Symbols
9	Arrows 3
10	CRV

Table 2. Icon Types

The following example uses the ICONSETS argument to add trafficlighting icons that provide a quick visual reading of profit-analysis data.

```
ods tagsets.tableeditor
    file="temp.html"
options(format_condition="iconsets,2,5"
        pivotrow="product_category"
        pivotdata="profit"
        pivotdata_fmt="#,###.##"
        pivotdata_caption="Profit
                           Analysis"
        pivot_grandtotal="no");
proc print data=sashelp.orsales;
run;
ods tagsets.tableeditor close;
```



Book3 - Microsof...

A1 Profit Analysis

	A	B	C	D
1	Profit Analysis			
2	Product_Category	Total		
3	Assorted Sports Articles	🟢 9,994,898.76		
4	Children Sports	🔴 2,417,119.78		
5	Clothes	🟡 9,208,375.44		
6	Golf	🔴 3,711,822.11		
7	Indoor Sports	🔴 1,481,330.76		
8	Outdoors	🟢 13,400,513.15		
9	Racket Sports	🔴 2,016,834.77		
10	Running - Jogging	🔴 2,300,666.19		
11	Shoes	🟡 8,889,545.98		
12	Swim Sports	🔴 727,868.6		
13	...ports	🔴 ...007		

Display 17. Using Trafficlighting Icons to Highlight Data

Creating Calculated Columns

Calculated columns are one of the most powerful concepts for pivot tables, and they are easy to implement with the tableEditor tagset. This feature provides the ability to specify a formula for calculations within a column. For example, you can estimate the number of items that will be returned based on the number of items you sell. If, historically, your returns average 5% of your sales, you can adjust the net profit to account for the adjusted 5% returns by creating a formula to calculate this information. You calculate the amount of the adjusted returns and the value of profits minus this historical return value.

Consider the following sample code:

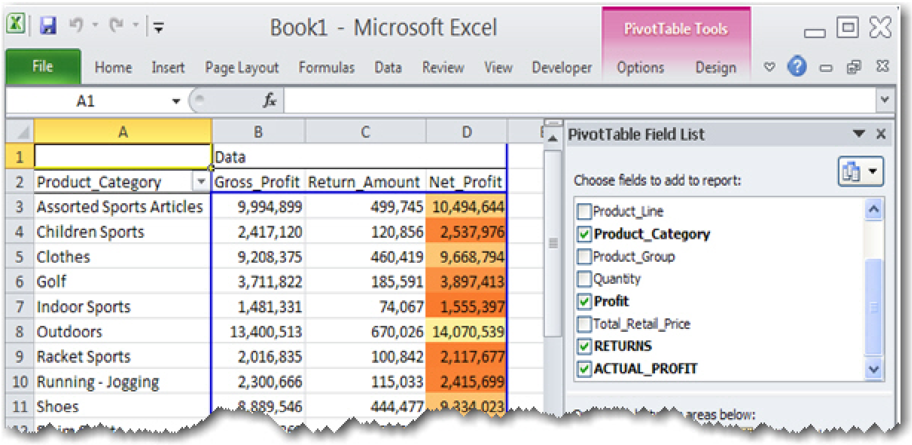
```
ods tagsets.tableeditor file="temp.html"
options(addfield="Returns =Profit*.05,Actual_profit =Profit*1.05"
        pivotdata="Profit,returns,actual_profit"
        pivotrow="product_category"
        pivotdata_fmt="#,###,##"
        pivotdata_caption="Gross_Profit,Return_Amount,Net_Profit"
        pivotdata_tocolumns="yes"
        format_condition=",colorscale,4"
        pivot_grandtotal="no");
```

(code continued)

```
proc print data=sashelp.orsales;
run;

ods tagsets.tableeditor close;
```

This code adds calculated columns by using the ADDFIELD= option. The argument for this option is the formula for the new fields. If you want to add multiple calculated columns, then each column and formula is separated by a comma. The new calculated column names should also be added, along with the variables you want to analyze, to the PIVOTDATA= option.

The two columns that are generated in this example are **`Returns`** and **`Actual_Profit`**. The **`Returns`** column uses the formula =`Profit*.05` to calculate an allowance of 5% of the profits. The **`Actual_Profit`** column uses the formula =`Profit*1.05` to calculate the adjusted profit after returns. The sample code also uses the PIVOTDATA_CAPTION= option to rename the fields from the default label **`Sum of`** ***`variable-name`*** to a more meaningful label, as shown below.



	A	B	C	D
1		Data		
2	Product_Category	Gross_Profit	Return_Amount	Net_Profit
3	Assorted Sports Articles	9,994,899	499,745	10,494,644
4	Children Sports	2,417,120	120,856	2,537,976
5	Clothes	9,208,375	460,419	9,668,794
6	Golf	3,711,822	185,591	3,897,413
7	Indoor Sports	1,481,331	74,067	1,555,397
8	Outdoors	13,400,513	670,026	14,070,539
9	Racket Sports	2,016,835	100,842	2,117,677
10	Running - Jogging	2,300,666	115,033	2,415,699
11	Shoes	8,889,546	444,477	9,334,023

- ☐ Product_Line
- ✓ Product_Category
- ☐ Product_Group
- ☐ Quantity
- ✓ Profit
- ☐ Total_Retail_Price
- ✓ RETURNS
- ✓ ACTUAL_PROFIT

Display 18. Calculated Fields Added to a Pivot Table

IMPROVING OVERALL FUNCTIONALITY OF YOUR WEB PAGES

This next section focuses on improving the functionality of web pages that you generate from a desktop machine. The discussion looks at printing output from web pages, but it also explains how to add table column headings to tables that span multiples pages.

Adding Repeating Column Headings to Tables That Span Multiple Pages

One highly visible issue related to printing from web pages is that column headers do not repeat from one page to the next. Unlike the case with Adobe's PDF format, which is designed for printing, output viewed in or printed from browsers does not include repeated column headings for tables that are split across pages. Typically, this is not a big problem if you just have a few pages. But if a table spans several pages, it is easy to forget which column is associated with which column headings.

The following example creates output that spans two pages. To ensure that the column heading repeats on both pages, the code incorporates a CSS style by using the HEADTEXT= option.

```
ods html file="temp.html"
                headtext='<style>thead {display:table-header-group} </style>';

proc print data=sashelp.prdsale;

run;

ods html close;
```

The output from this procedure generates a total of 34 pages as you can see in the top right corner of the following display. The HEADTEXT= option causes the procedure to print column headers on pages 2 – 34.

When you generate output with the ODS HTML destination, a horizontal rule tag displays a logical break between each procedure's output. This tag provides a visual break in the output in the browser, and new procedures are started at the top of a new page, This behavior occurs because of the **`pagebreak-after:always`** style property within the horizontal rule tags instructs the browser to begin the output from a new procedure on a new page.

However, if you want to generate fewer pages, you can remove the page break with the PAGEBREAK=No tagset option. Starting with SAS 9.2, this option is added to the ODS destination statement. Prior to SAS 9.2, you had to use a style definition to remove the page break.

```
ods html file=”nobreak.html”  options(pagebreak=”no”);
proc print data=sashelp.class;
run;
proc print data=sashelp.class;
run;
ods html close;
```

CONCLUSION

In this age of computer information, web pages offer so much flexibility and power for presenting business and personal data. But users are not always aware of the how to use current technology to take their web pages to a whole new level of usability and functionality. This paper discusses methods for enhancing your web pages using the latest techniques in the SAS Output Delivery System along with the latest advances in cascading style sheets and HTML5. The discussion takes you across device boundaries, illustrating techniques specifically for output on mobile devices, for output on both mobile and desktop devices, and for output specifically for desktop devices. Once you understand these techniques, you have the ability to boldly take your web pages where they have never gone before!

RECOMMENDED READING

Apple Inc. (2013). Safari Dev Center. Available at **developer.apple.com/devcenter/safari/index.action.**

Eberhardt, Peter and Louanna Kong. 2012. "The Armchair Quarterback: Writing SAS® Code for the Perfect Pivot (Table, That Is)." *Proceedings of the SAS Global Forum 2012 Conference*. Cary, NC: SAS Institute Inc. Available at **support.sas.com/resources/papers/proceedings12/146-2012.pdf**.

jQuery Mobile (2013). Available at **jQuerymobile.com/.** Accessed on February 21, 2013.

Parker, Chevell. 2010. "Using SAS Ouptut Delivery System (ODS) Markup to Generate Custom PivotTable and PivotChart Reports." *Proceedings of the SAS Global Forum 2010 Conference.* Cary, NC: SAS Institute Inc. Available at **support.sas.com/resources/papers/proceedings10/003-2010.pdf.**

SAS Institute Inc. 2013 "Base SAS: ODS Markup Resources." Available at **support.sas.com/rnd/base/ods/odsmarkup/index.html.** Accessed on February 21, 2013.

Smith, Kevin D. 2011. "Unveiling the Power of Cascading Style Sheets (CSS) in ODS." *Proceedings of the SAS Global Forum 2011 Conference.* Cary, NC: SAS Institute Inc. Available at **support.sas.com/resources/papers/proceedings11/297-2011.pdf.**

Zender, Cynthia L.. 2009. "CSSSTyle: Stylish Output with ODS and SAS® 9.2." *Proceedings of the SAS Global Forum 2009 Conference.* Cary, NC: SAS Institute Inc. Available at **support.sas.com/resources/papers/proceedings09/014-2009.pdf**.

ACKNOWLEDGMENTS

I would like to thank Susan Berry for her editing contributions to this paper.

CONTACT INFORMATION

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

Chevell Parker
SAS Institute Inc.
Cary, NC
E-mail: support@sas.com
Web: **support.sas.com**