Paper 2670-2018

Take a Dive into HTML5

Brandon George and Nathan Bernicchi, Grand Valley State University and Spectrum Health Office of Research Administration; Paul Egeler, Spectrum Health Office of Research Administration

ABSTRACT

The SAS® Output Delivery System (ODS) provides several options to save and display SAS graphs and tables. You are likely familiar with some of these ODS statements already, such as ODS RTF for rich text format, ODS PDF for portable document format, and ODS HTML for hypertext markup language.

New to SAS 9.4, users have several additional ODS statements to choose from; among them is ODS HTML5. The ODS HTML5 statement differs from ODS HTML in that it uses HTML version 5.0 instead of HTML version 4.01. As such, the user can now take advantage of several new features introduced to the language. ODS HTML5 also uses a different default graphics output device, trading out the old standby portable network graphics (PNG) for the up-and-coming scalable vector graphics format (SVG).

Throughout the paper, we will look at some of the differences between ODS HTML and ODS HTML5, with a focus on computer graphics. The paper also begins with a short primer on technologies such as HTML, CSS, and computer graphics. We will also demonstrate a few ways that you can enhance your reports using some ODS tricks. Our goal is to convince you to switch to ODS HTML5 and start using SVG as your graphical output of choice.

INTRODUCTION

In SAS®, you have numerous options for how to output your graphs and tables. You can create a rich text format (RTF) document, a PDF document, or even an HTML static webpage, just to name a few! Each output destination has its advantages and disadvantages. For example, HTML documents are excellent for sharing reports (not just on the web) because you know that your audience is most likely going to have a browser to view its contents. HTML documents also give you a lot of control over how your content is displayed, navigated, and integrated with other documents. And now, SAS has just made designing beautiful and functional HTML reports better with their new output delivery system, ODS HTML5.

HTML was first introduced back in the early 1990's and has gone through several versions, the most recent being HTML5, which was released in 2014 (W3, 2014). HTML5 adds a lot of functionality to the HTML specification—such as embedded audio and video! It also represents a philosophical shift in how HTML documents are created (W3, 2014). This cleans up a lot of problems that web developers used to face with older HTML specifications. SAS followed suit and added the ODS HTML5 destination to SAS 9.4 to take advantage of the recent HTML specification update. ODS HTML, which follows the HTML 4.01 specification, is still available but is a separate ODS destination.

You are probably familiar with ODS HTML since it has been the default delivery destination in the SAS Windowing environment since SAS 9.3. You might also be familiar with ODS HTML5 if you use SAS Studio. You might think that they are basically the same since both output HTML documents in the same HTMLBlue style by default. Indeed, plain vanilla ODS HTML output has much the same look and feel as plain vanilla ODS HTML5 output. But underneath, the two documents are constructed differently. This allows you to take advantage of some great new features that were not previously available to you.

One of the biggest changes that came with ODS HTML5 was the way that graphics are output. The default graphics device for ODS HTML is portable network graphics (*.png*). In ODS HTML5, the default graphics device is scalable vector graphics (*.svg*). This single change can have a huge effect on the image quality of your plot outputs and how they are used. We will explain some important concepts that will help you get the most out of this new image format.

We will begin the paper by explaining the basic background of HTML and its related technologies and give you a brief primer on computer graphics file formats. We will go into some of the differences between

HTML 5 and its older implementation. We will then show you how you can make the most of your ODS HTML5 documents.

It's time to dive in!

A LITTLE TECHNICAL BACKGROUND

This paper contains some tech jargon that might be new to you. We're going to be talking about web technologies, including HTML and CSS, as well as graphics file formats. You are likely familiar with these technologies as an end-user, but you might not have gotten into the nitty-gritty of graphics outputs or looked under the hood of your favorite website before. Therefore, let's take just a little time to cover the basics of some of the technologies we're going to talk about.

WHAT IS HTML?

Hypertext markup language (HTML) is the language of web pages. It is a language that consists of tags that give structure to a document. The document can be thought of as a tree, where each node (denoted by a tag) can branch into zero, one, or many descendent nodes. Following the tree analogy, we use familial terms to denote the relationships of nodes to each other. A direct descendent of a node is its child node; likewise, its immediate ancestor is its parent node. Nodes at the same level are siblings. The top-most node (highest ancestor) is called the root node.

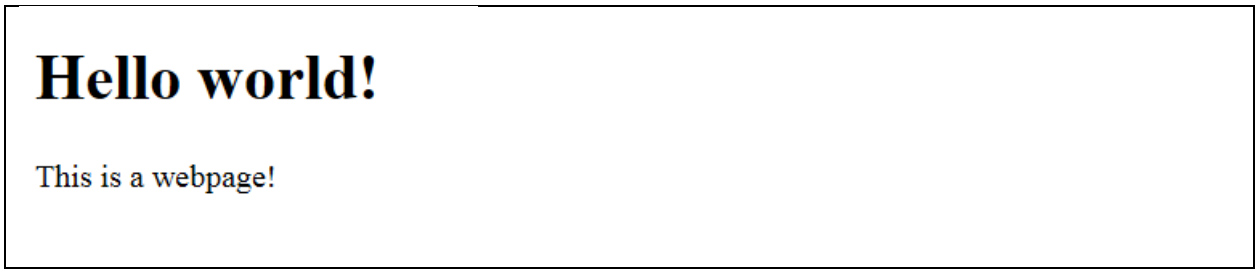
These nodes provide the semantic context for all your content, such as the document title, sections and headings, paragraphs, images, tables, and hyperlinks. These nodes also contain information that the browser does not directly render, such as metadata, scripts, and styling instructions.

Tags are enclosed in angle brackets (< and >) and usually come in pairs, called "opening" and "closing" tags. Although XML (*e.g.*, XHTML) requires that all tags be closed, HTML5 has looser rules with regard to some tags, so not all tags are required to be closed. Tags must be nested so that a hierarchal structure is maintained throughout the document. Let's take a look at this basic website to explore those concepts:

Code:

```
001 <!DOCTYPE html>
002 <html>
003   <head>
004     <meta charset="UTF-8"/>
005     <title>An example webpage</title>
006   </head>
007   <body>
008     <h1>Hello world!</h1>
009     <p>This is a webpage!
010   </body>
011 </html>
```

Result:



The table below goes over the example above line-by-line so we can understand what it all means.

Line	Explanation
001	The first line declares the document type as HTML (it is implied that we are specifying HTML version 5)
002	This tag begins the HTML document. It is the root node. All nodes within the document will be descendants of this node.
003	This begins the header section of the HTML document. It is the first direct descendant (child) of the `<html>` node.
004	The metadata tag contains an attribute that defines the character set used in this document. This an example of a self-closing tag: it does not need a closing `</meta>` tag because the `/` character is present at the end of the opening tag.
005	This is the title that will show up in the title bar of your browser. Note that the opening and closing tags are on the same line in this example.
006	This is a closing tag for our `<head>` tag on line 003.
007	This line begins the body of the document. This is the second child of the `<html>` tag and a sibling of the `<head>` tag. Content within the body may be (but is not always) rendered by the browser.
008	This is a heading level one. Think of it as the main title. Headings are usually rendered with larger font size.
009	The `<p>` tag indicates a paragraph node. Closing this node (with a `</p>` tag) is optional.
010	This closes the `<body>` node that was opened on line 007.
011	The `</html>` closing tag indicates the end of the HTML document.

WHAT IS CSS?

Whereas HTML provides the content and semantic structure of a document, cascading style sheets (CSS) are used to determine the styling, formatting, and layout used when your browser renders the web page.

Basically, CSS allows you to identify different elements within your document and assign different properties to them. You can define styles for specific elements or element types. You can even assign styles based on node attributes and classes. For example, you might want all text enclosed within *emphasis* (`<em>`) tags to be bold and red. You could define that like this:

```
em {
  font-weight: bold;
  color: red;
}
```

Similarly, you might want text in a specific node to be twice as big as the rest of the text on the page. You can use the CSS selector `#` to identify the node with the *id* property that matches like so:

```
#my-node {
  font-size: 2em;
}
```

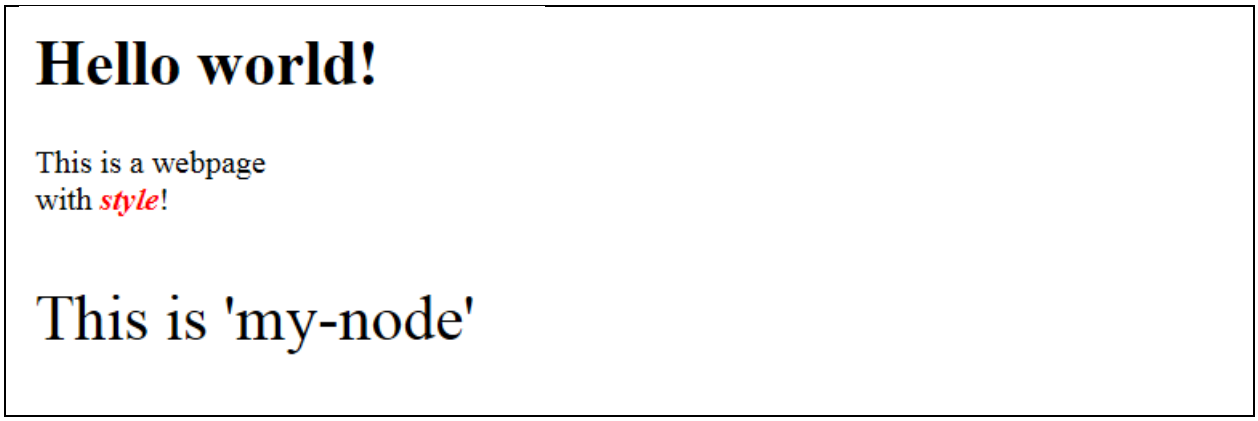
Then the text within the node that has the ID *my-node* will have double-sized font.

When embedding CSS in an HTML document, it is customary to do so in the *header* of the document within a `<style>` tag (although you can also assign styles to specific elements as attributes). Adding these concepts to our example webpage above, we will just add a little bit of code:

Code:

```
001 <!DOCTYPE html>
002 <html>
003   <head>
004     <meta charset="UTF-8"/>
005     <title>An example webpage</title>
006     <style>
007     em {
008       font-weight: bold;
009       color: red;
010     }
011     #my-node {
012       font-size: 2em;
013     }
014     </style>
015   </head>
016   <body>
017     <h1>Hello world!</h1>
018     <p>This is a webpage
019     <br>with <em>style</em>!
020     <p id="my-node">This is 'my-node'
021   </body>
022 </html>
```

Result:



As you can see, we've added a little bit of flair to our basic HTML document. Line 006 nested in the header node begins our style node. Within the style node, we can use CSS syntax to define how elements of the webpage are rendered. At line 014, we close the style node and are back to HTML syntax. We also show the use of a `<br>` tag at line 019 that inserts a line break within a paragraph. Like the `<p>` tag, this `<br>` tag does not need a closing tag. On that same line, we also introduce `<em>` opening and closing tags around the word *style*, which renders the font as italicized by default and is also bold and red thanks to our CSS above. Finally, we show how to add attributes to tags using name-value pairs on line 020. In this case, we are giving the `id` attribute the value `"my-node"` for that specific paragraph element. This helped us uniquely identify the `"my-node"` paragraph using a CSS selector so that we could change the font size.

You can embed CSS into your HTML document like we saw above, or you can create a link to a local CSS file or URL in your HTML documents. Linking to a common CSS file has the advantage of allowing you to have consistent styling on all of your HTML documents; you only have to worry about making changes in one place rather than in each HTML document individually. This also makes your files smaller

by allowing you to define your styles in one centralized spot rather than having to repeat all that information in each document. With that said, it is nice to be able to embed CSS in your HTML document so that the document is self-contained. For example, if you are sending an HTML document as an email attachment, you don't have to worry about whether the recipient has the correct CSS files or a constant connection to the internet!

The World Wide Web Consortium (W3C), which maintains the specifications for HTML and CSS, encourages the use of CSS for controlling presentational aspects of your webpage. Meanwhile, your content and its semantic relationships are stored in the HTML document itself. A big part of the HTML5 specification is about reinforcing that paradigm.

WHAT'S NEW IN THE HTML5 SPECIFICATION?

The HTML5 specification is largely similar to its most recent predecessors, HTML4 and XHTML. One major component of the change is that it adds some new tags, retires some old ones, and changes the meanings of a handful of others (W3C, 2014). It also provides web developers with more tag attributes and changes some rules and syntax. For the most part, these changes contribute to cleaner, more standardized documents. But some of these changes also result in some cool functionality we haven't previously seen with pure HTML.

You probably are not interested in learning the minutiae of all that's new and different in HTML5 unless you want to go on to do some more web development outside of generating SAS reports with ODS HTML5. However, there are a few important application programming interfaces (APIs) and other features that have been added that you will want to be aware of:

- **Multimedia Playback:** Prior to HTML5, you needed an external plugin for your browser in order to play movies or music. Some of you might remember Adobe Flash, which was a common plugin of choice for that kind of thing. Now with HTML5, audio and video playback can be handled natively within the HTML document and customized with JavaScript. Starting in the third maintenance release of SAS 9.4, this API has been brought to you so that you can embed audio and video into your SAS reports with a few data step commands.
- **Scalable Vector Graphics (SVG):** SVG is a graphics format that is perfect for the web. It is compact, scalable, and easily read by humans and computers alike. Its data format is based on XML, so its syntax fits neatly within an HTML document. We will go into SVG in greater depth in just a moment.
- **Mathematical Markup Language (MathML):** HTML5 supports MathML, which means that you can render professional-looking mathematical formulae in your HTML document using natively-supported syntax. MathML is also based on XML syntax. (W3C, 2003)

HTML5 also represents a philosophical shift as it codifies a lot of the advice the W3C has been giving around using HTML for structural content and CSS for presentational rules. These changes are also reflected in the HTML code generated by ODS HTML5.

A 5-MINUTE INTRO TO COMPUTER GRAPHICS FILE FORMATS

Think of how many graphics you see out on the World Wide Web. You have logos, pictures, icons, and more. All of those graphics are stored in one of two types of files: 1) raster graphics files or 2) vector graphics files. You are probably familiar with raster graphics if you've ever used a digital camera or made a picture in MS Paint, but you may be less familiar with vector graphics. We hope that is about to change!

Here is what you need to know to understand the advantages of each file type and how they are used in practice.

Raster Graphics

A raster graphic is an array of data organized into a grid of *pixels*. Each element of the data represents point on the screen which is represented by a color defined by the data element's value. Hence we get the word "pixel," which is a portmanteau combing the words "picture" (*pix*) and "element" (*el*).

You set the number of pixels that a file contains when you save it, so resizing it can cause issues. These graphics give you a pixelated, or blocky, appearance when you zoom in. This also means that you can

lose detail in the smaller features of an image unless you make your image file large enough so that it has enough pixels to adequately represent the image.

There are dozens of different raster graphics file types, and they all have their own purpose. Each format has its own strengths and weaknesses which will affect how the displayed graphic looks, how much disk space it will take, what software is able to display it, and even whether it is considered publication quality. Carefully consider which file type, file size, and color space you choose when creating plots so that your images are represented appropriately for their intended medium and purpose!

Raster Graphics File Types

There are a few file types you should definitely know. You have probably made plots before in the portable network graphics (*.png*) format (a common default format for SAS ODS). Meanwhile, you have probably taken pictures which were saved in the Joint Photographic Experts Group (*.jpg* or *.jpeg*) format. A few other types that might be familiar are bitmap (*.bmp*), tagged image file (*.tiff*), and graphics interchange format (*.gif*). We will go into detail about a few of the file formats below.

Portable Network Graphics (PNG)

Portable network graphics (PNG) files are the default graphical output in SAS ODS HTML. These files use lossless compression. This means that the graphic is compressed, but all of the original information is still available from the original graphic. In other words, you can reconstruct the image exactly as it was in the original file. Lossless compression results in a better quality image, but the file sizes can be rather large when compared "lossy" compressed files. PNG supports a lot of different ways to represent colors, such as grayscale, pseudocolor (indexed color), and Truecolor (24-bit color) plus an optional alpha channel for transparency. All mainstream browsers and image viewers support PNG files and they offer an acceptable mix of file size and graphic quality for web and general purpose images.

Joint Photographic Experts Group (JPEG)

Unlike PNG files, JPEG uses "lossy" compression. This compression allows you to greatly decrease the size of graphic file. However your compressed file may differ from the original file and there is no way of returning to that original file. Often times you can choose what degree of compression you want, which means you can choose how much quality you want to lose in the image (how small you want the file to be). The loss in data is often not seen by the human eye, but that is not the case in every situation. This file type is commonly used by digital cameras. It supports Truecolor, but does not support an alpha channel for transparency.

Graphics Interchange Format (GIF)

This type, like PNG, uses lossless compression which means the original file is not lost during the compression stage. The difference between GIF and PNG is that GIF has a much smaller file size in comparison to other lossless compression files. GIF files are compressed by reducing the number of colors down to 256 and by replacing multiple similar colors into one. For example, instead of storing five different blues for a certain graphic; it will only store one blue. Obviously this isn't the format of choice for any colorful graphic, but it does work well for graphics with few colors. Another attribute of GIF is that you can include animation with your graphics, which can be an advantage over other file types with many graphs.

Image Compression and Anti-aliasing

Image Compression

One main difference between these file types is the way the files are compressed. As we mentioned before, file formats such as PNG use lossless compression, meaning that the original raster image can be reconstructed exactly. Meanwhile JPEG images use lossy compression, which results in a loss of data. This means that your JPEG images will be approximations of the original content, resulting in unattractive *artifacts* where parts of the image look fuzzy or blocky.

Let's take this example of an image of a SAS programmer diving into HTML5 for the first time (Figure 1). The file has been saved in PNG format with lossless compression with and without using anti-aliasing techniques (left and center). The same image has also been saved as a JPEG file with high lossy compression (far right). Note the fuzzy, discolored edges compared to the other two PNG images on the left. These are compression artifacts. Figure 1 also zooms in to show how the images differ close up.

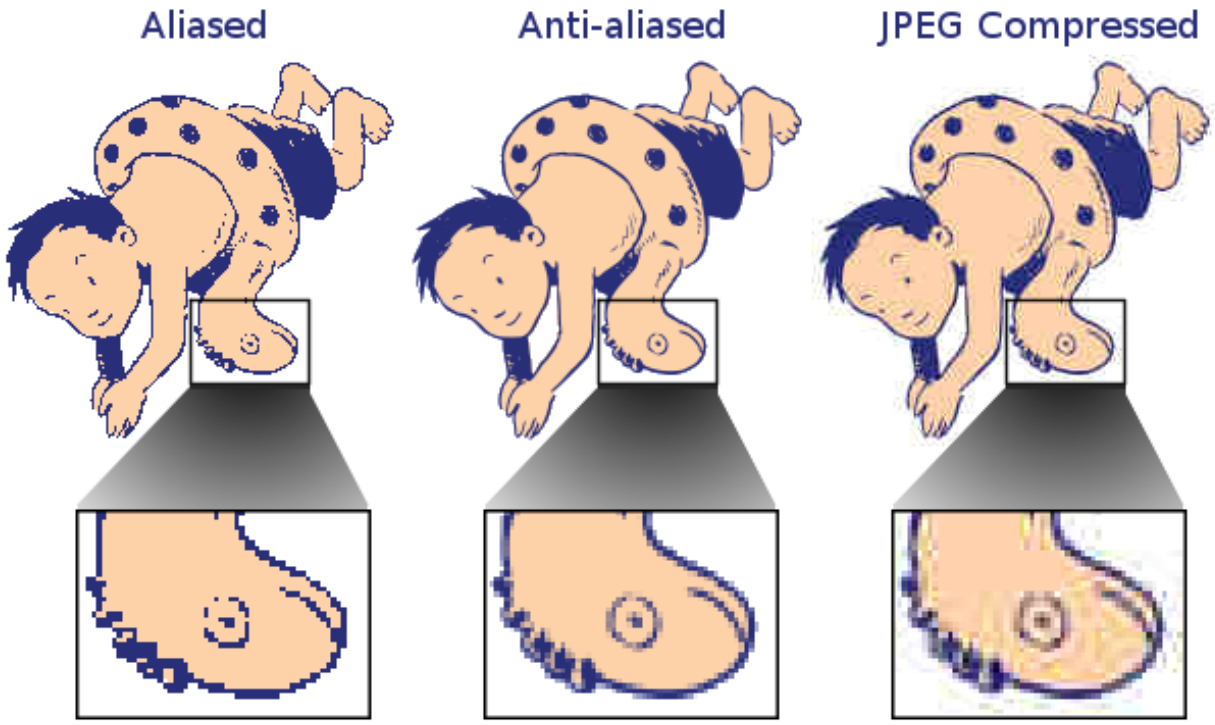


Figure 1. Kid Diving into HTML5.

Anti-Aliasing

Raster images sometimes don't look good because your eye can see the individual pixels that make up a curve or diagonal line. The jaggedness that you see is a result of *aliasing*. You see it even more when the raster image is blown up larger than its original resolution. This is called *pixelation*.

SAS makes your statistical graphics raster images look a little better by using anti-aliasing. Anti-aliasing blends image components so that pixels adjacent to the curve will appear smoother. This gives your image the illusion of a higher resolution. However, the price you pay is that the blending can muddy detail on some images. Figure 1 shows the same zoomed-in section of our diving SAS programmer with and without anti-aliasing. You control this as an option in your ODS GRAPHICS statement. It is on by default and we recommend that you leave it on!

Vector Graphics

Unlike raster graphic files, vector graphic files contain instructions of how to draw a given graphic rather than the *rasterized* image data itself. This set of instructions tells the computer where each component starts and ends, as well as curves along the way. You can even use it to specify where text should be drawn. This allows the computer to rasterize the image on the fly, meaning that panning and zooming will re-draw the image.

This can result in better image quality and higher fidelity to detail. Your biggest advantage to using vector graphics over traditional raster graphics is the idea of resizing the image. When you resize a vector graphic the paths are redrawn and you do not lose image quality in the process. When blowing up a

raster graphic, you get pixelation (explained above). This look appears unprofessional and can even cause issues when submitting a manuscript to a publisher.

Let's revisit our image of the SAS programmer to demonstrate the advantages of using vector graphics. This image is stored as a vector graphic. We have rasterized it at several zoom levels to show that detail is not lost (Figure 2). The computer redrew the image at each zoom level to maximize quality.

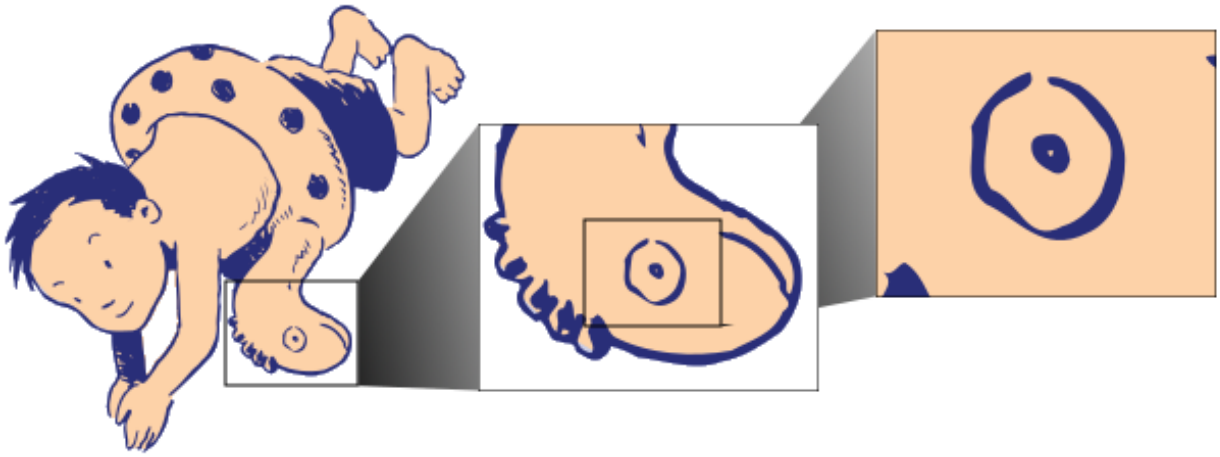


Figure 2. Zooming in to show image quality is conserved.

Another bonus of vector graphics is that they are often smaller files in relation to a similar raster graphic file. Images with relatively few graphical elements can be stored as just a few "drawing instructions" in the vector file format, rather than as a large array of pixels.

There are several common vector image file formats, such as Encapsulated PostScript (*.eps*), Adobe Illustrator (*.ai*), and scalable vector graphics (*.svg*). The file format that is probably most relevant to you as a SAS programmer is the SVG format, so we will focus on explaining that one in detail.

Scalable Vector Graphics (SVG)

The scalable vector graphics (SVG) file format is a very common vector graphics format. It is defined by the World Wide Web Consortium (W3C) and is an open standard. The data are stored in e**X**tensible **M**arkup **L**anguage (XML), which means that the information can be embedded directly into an HTML document or stored as a standalone file. This also allows the graphic to be both human and machine readable. You can even create and edit SVG files in a simple text editor!

These files are comprised of three components: 1) vectors, 2) images, and 3) text (W3C, 2011). This means you can mix and match the three components to suit your needs. For example, you may want to create a plot with your company's logo as a raster image, but have the plot itself rendered as vector images, and have the axis labels as actively rendered text. SVG graphics are more "alive" than their raster counterparts; interactive and dynamic graphics (such as animations) can be done relatively easily.

Because SVG graphics are built on XML, they can be integrated tightly with the HTML document, styled using CSS, and brought to life using scripting (like JavaScript). This means you can do things like change components of the image on mouse-over or add tool-tips. Scripting opens up a wide world of dynamic content display and interactivity that is well beyond the scope of this paper.

WHAT'S NEW IN ODS HMTL5

So now that we have covered some of the more technical background of web technologies and graphic files, we can dive into what might be most important to you as a SAS user in ODS HTML5. There are numerous changes and additions made within ODS HTML5 over what is available in ODS HTML, so we will not be covering every single difference between the two output delivery systems.

One of the bigger changes that SAS made in ODS HTML5 was how graphics are handled. ODS HTML5 now has the ability to completely encode your plots right into the HTML file. In the past, with ODS HTML,

you had keep track of multiple files for a report—the report itself as well as plots and graphics. This meant that you had to send multiple files when emailing reports and dealing with broken image links when moving a report from place to place. Now with ODS HTML5, your graphics can be associated with your reports in several ways. In fact, the default is that the graphics are encoded directly into the HTML document so you will have standalone HTML reports that can be easily shared with colleagues (SASf).

Another change in ODS HTML5 is the default graphics output format. In ODS HTML, the default graphics device was PNG. Now with ODS HTML5, the default graphics device is SVG (SASf). In laymen's terms, the major advantage of using SVG is that you can resize the image without worrying about the blown up image looking fuzzy or jagged. Figure 2 above provides a good illustration of this. We will paddle out to explore this more in the MAKING THE MOST OF ODS HTML5 section below.

SAS has also incorporated the HTML5 specification's ability to embed audio and/or video content right in your HTML output document. Starting in the third maintenance release of SAS 9.4 (SAS 9.4M3), you can put audio and video files right in your HTML document using the Report Writing Interface. The applications with this are virtually endless. For example, you could add in an audio file that contains your voice explaining important output or a video that helps explain what a certain output means. We will dive more into depth on this idea in the next section.

MAKING THE MOST OF ODS HTML5

We have talked about many of the new additions and options you have with ODS HTML5. So, how can you use these changes to make the most of your SAS experience? In this section, we will highlight some key ideas to brush up on from outside ODS HTML5, as well as show you how to implement all the new tools you are offered within ODS HTML5.

ACCESSIBILITY

Accessibility refers to making the content of a document accessible for everyone, including people with disabilities. You have to think about the part of your audience with sight impairments, hearing impairments, and/or other impairments. Having an accessible document will allow these folks to navigate and understand your document more easily. According to SAS, ODS HTML5 provides the best accessibility among all ODS destinations. If you want your document to be readable and accessible by everyone, using ODS HTML5 and its accessibility features will allow you to make this happen.

The first and most important step to accessibility is having a good design (SASc). In an HTML document, this means both a clean look to the content as well as clean HTML code working in the background. Removing visual clutter, choosing colors with care, and using high contrast themes can help people with low vision or other sight impairments. Meanwhile, well-structured and annotated HTML can be read and understood more easily by screen reader software.

Here are a few specific examples of how ODS HTML5 can help make your reports more accessible (SAS):

- The ODSTEXT procedure within ODS HTML5 allows to specify tags such as heading tags (`<h1>`, `<h2>`, etc.). This will help keep your document well organized and readable.
- Use of RWI to enhance accessibility using IMAGE, TABLE_START, LAYOUT_ABSOLUTE, LAYOUT_GRIDDED, REGION Gridded, REGION Absolute.
- Using DESCRIPTION to add alternate text in various statements.
- Custom table of contents (more below)
- Adding style (more below)
- Inline SVG graphics (more below)

We don't have enough space to go into great detail about how to maximize the accessibility of your document. As noted above, we will highlight some of these features in the following sections. Please consult the RECOMMENDED READING section for a link where you can find details.

CUSTOM CSS AND STYLES

To give you a slight refresh on what was covered in the WHAT IS CSS? section, remember that CSS is what controls the styling, format, and layout of the given document. SAS allows the use of CSS three different ways: 1) embedded, 2) inline, and 3) external (SASa).

- **Embedded:** Style, formatting, *etc.* is defined in the header of the HTML file. SAS allows you to generate embedded CSS in your HTML output. This allows you to make standalone HTML files whose styling is not dependent on an internet connection or the presence of a styling file.
- **Inline:** Much like embedded CSS, inline CSS can generate standalone HTML files. However, in this case, styles are defined tag-by-tag rather than in the header of the HTML document. This type of styling is discouraged unless there is a good reason to do so.
- **External:** You can link to CSS files based on relative or absolute path on your file system. You can also link to CSS files on the internet. This has many advantages, such as consistency across reports and the reduction of file size. However, the styling of these reports can be lost if the CSS file is unlinked somehow.

These concepts are not unique to ODS HTML5. The use of these CSS styling techniques is available in several other output delivery systems, such as ODS HTML and even ODS PDF. However, they are so powerful that it is worth reviewing them here.

The following code shows how you can use the `headtext=` option to create embedded CSS styles. This code will affect all content within a `<td>` tag. Specifically, it will change the font color within the body of the table. The `<td>` tag denotes table data, so in this case the data cells of the PROC PRINT output table will be affected. Everything else will be styled based on the built-in “minimal” ODS style. The code can be seen below:

```
filename filedir "c:\Users\&sysuserid\Desktop";
ods html5
  path=filedir
  file="example_embedded_css.html"
  style=styles.minimal
  headtext='<style>td {color: red;}</style>';

proc print data=sashelp.fish
  style(table)={prehtml="<p style='text-align: center;'>Fish</p>"}
  style(data)={htmlstyle="font-weight: bold;"};
run;

ods html5 close;
```

Note in the code above that we’ve also snuck in some inline styling using the `style` option in the procedure statement. In this case, it does not make a lot of sense to use inline styling to change the font weight (since we could include that in our embedded CSS). However, there may be times when it is appropriate, like when we want to quickly center a single paragraph.

You can also embed CSS into a report using the `cssstyle` option in your `ods html5` statement. This option allows you to reference a CSS file while creating a report rather than writing CSS in the SAS program. The stylings are all embedded in the header of the outputted report, so you can share a single HTML document instead of having to send both the HTML and CSS document for your stylings to be preserved.

The TEMPLATE Procedure is another means for creating customized CSS that can be embedded in the document. You can use it to assign HTML classes or further customize styles. The TEMPLATE procedure is an extremely powerful procedure that can be used to fine-tune stylings of reports. It allows you to write in CSS or more SAS-like syntax that will be translated into CSS.

You can create a report that links to a separate CSS file using the `stylesheet` option in your ODS HTML5 statement. The CSS will not be embedded in the report at all, but will point to the separate file. The advantage of linking to external CSS files is that styles can be easily and uniformly defined for a set of reports. Whether you have company standards or just a style you like, reproducing this style is easy with the use of external CSS. This code will create an HTML document that will link to a CSS file hosted by Google:

```
ods html5
  file="example_external_css.html"
  stylesheet=(URL="https://www.google.com/css/maia.css");
```

One major disadvantage of linking to an external CSS file is that you need to make sure that your recipient has access to the same CSS file in the same file path on their system or a link to the internet to access the URL. Otherwise the styling information will not display!

GRAPHICS AND PLOTS

The new graphics features are a big selling point of ODS HTML5. Not only does ODS HTML5 bring the SVG format to the forefront, but there are some new options on how your graphics are stored and displayed. For example, now your vector and raster images can be encoded directly into your HTML document, so it can act as a standalone file.

Graphics in SAS are created two different ways: either with traditional SAS/GRAPH techniques or with the newer statistical graphics (SG) outputs. In the traditional SAS/GRAPH system, graphical output is controlled using OPTIONS and other global statements (GOPTIONS, AXIS, LEGEND, etc.). Meanwhile, SG outputs are controlled with the ODS GRAPHICS statement and statements or options within individual procedures. We will show a use-case for both.

We like to use the statistical graphics procedures offered by SAS to get great-looking outputs with the least amount of coding. By turning ODS GRAPHICS ON, you gain access to several statistical graphics procedures and plot outputs that look great with a minimal amount of configuration. It is important to point out that setting the graphics device (*i.e.*, using the ODS HTML5 or GOPTIONS statements) will only change the output format of traditional graphics under the SAS/GRAPH procedures. The code below will specify that you want ODS GRAPHICS turned on and that you want all graphics to be outputted in an SVG format:

```
ods graphics on / imagefmt=svg;
```

In ODS HTML5, you are able to control how you want your graphics file to be linked to the HTML file. You have the ability to choose whether you want the image data to be encoded within your HTML file or have the graphic written to a standalone file. Within your ODS HTML5 statement, you have several suboptions to control this: either `options(svg_mode=)` for SVG graphics or `options(bitmap_mode=)` for raster graphics. Using INLINE for either suboption will encode the data directly into the HTML report. It will be entirely contained within the HTML document. This is the default when displaying SVG graphics, but needs to be specified for raster graphics. The code below shows how to INLINE a PNG file into an HTML document:

```
/* OPEN ODS HTML5 */
ods html5
  path=filedir file='example_inline_PNG.html'
  options(bitmap_mode='inline');

/* TURN ON ODS GRAPHICS AND SET IMAGE FORMAT TO PNG */
ods graphics on / imagefmt=png;

/* PLOT USING STATISTICAL GRAPHICS PROCEDURE */
proc sgplot data=sashelp.fish;
    scatter x=height y=weight/group=species;
run;
```

```
/* CLOSE ODS DESTINATION */
ods html5 close;
```

Sometimes the INLINE format can be cumbersome since it is not supported by all browsers or email clients and makes your graphic less portable. Therefore, it might make sense to save the image file separately using one of the other methods available to you. Using IMG for either the `svg_mode=` or `bitmap_mode=` suboptions will create an `<img>` tag in your HTML document that will link to a standalone image file. When you do this, SAS will save those graphics in the same location specified with your path statement in the ODS HTML5 line. So, the following code will create the same HTML5 as above, but will also create a standalone PNG file instead of "baking in" the image data into the HTML document:

```
/* OPEN ODS HTML5 */
ods html5
  path=filedir file='example_img_PNG.html'
  options(bitmap_mode='img');

/* TURN ON ODS GFX, SET IMG FORMAT TO PNG, AND SPECIFY IMG NAME */
ods graphics on / imagefmt=png imagename='example_fish_plot';

/* PLOT USING STATISTICAL GRAPHICS PROCEDURE */
proc sgplot data=sashelp.fish;
    scatter x=height y=weight / group=species;
run;

/* CLOSE ODS DESTINATION */
ods html5 close;
```

Animated/Interactive Charts

You have the ability to include SVG animated or interactive charts and plots, too. Data visualization is an extremely popular topic these days and having the ability to create these types of plots can be a way to improve your data visualization skills.

In our example fish dataset, we want to show a bar chart for each species that shows the distribution of weight. Printing out a graph for all seven species of fish may result in a confusing and cluttered document. With the use of the interactive charts, you can make a single plot that will cycle through all seven species of fish and give you the appropriate corresponding graph. This will allow your document to be much cleaner and even help with comparisons between the species as the chart cycles through each one.

We use traditional output from SAS/GRAPH procedures in the code below to show how to make an animated plot. In this example, we will have each SVG up for 3 seconds, the new SVG will take 1 second to fade in, the loop will go continuously, and the SVGs will overlay each other. In this case, we have also made the SVG file INLINE with our HTML document, meaning that the data to create the animation is built into the HTML rather than in a standalone file. The code can be seen below (Mincey):

```
/* CREATE FORMATS AND DATA */
proc format;
value length_ranges
  low - 15 = '0-15'
  15 <- 30 = '16-30'
  30 <- 45 = '31-45'
  45 <- 60 = '46-60'
  60 <- high = '60+'
;
run;
```

```
data ranges;
input range $ @@;
cards;
0-15 16-30 31-45 46-60 60+
;
run;

proc sql;

create table fish_lengths as
  select
    a.species, a.range, coalesce(b.n_fish,0) as n_fish
  from (
  select
    species.species, ranges.range
    from (select distinct species from sashelp.fish) as species, ranges
  ) as a
  left join (
    select
      species,
      put(length1, length_ranges.) as range,
      count(*) as n_fish
    from sashelp.fish
    group by 1, calculated range
  ) as b
  on a.species = b.species and a.range = b.range
  order by 1,2
  ;

quit;

/* SET THE SIZE OF THE SVG GRAPH */
options svgheight='400px' svgwidth='600px';

/* SET THE DEVICE= OPTION TO SVG */
goptions reset=all device=svg;

/* SET THE SYSTEM OPTIONS TO CONTROL THE SVG ANIMATION */
options printerpath=svg animate=start animduration=3
  svgfadein=1 animloop=1 animoverlay nodate nonumber;

/* OPEN ODS HTML5 DESTINATION */
ods html5
  path=filedir
  file="example_animated_plot.html"
  options(svg_mode="inline");

/* USE PROC GCHART TO CREATE THE FINAL ANIMATED GRAPH */
title "Finland's Lake Laengelmavesi Fish Catch Data by Species";
proc gchart data=fish_lengths;
  by Species;
  vbar range / sumvar=n_fish sum raxis=axis1 maxis=axis2 patternid=midpoint
width=15;
  axis1 order=(0 to 40 by 5) label=('Frequency');
  axis2 label=('Length Range');
run; quit;
options animate=stop;
```

```
ods html5 close;
```

This is a very simple application of the animated charts. You can also create interactive charts using ODS HTML5. These are much more complicated but can be very user friendly and efficient in getting information across. You can imagine how many different animated/interactive charts that can be made ranging from a simple “cycled bar chart” like this to incorporating interactive behavior such as tooltips. As with many features in ODS HTML5, the applications are endless.

EMBEDDING AUDIO AND VIDEO CONTENT

Embedding audio and video directly into your HTML output may be one of the more interesting new additions to ODS HTML5. The audio and video is handled by the browser, which means you don’t need to work with any special plugins or scripts to seamlessly integrate audio/video content into your reports.

You can embed audio and video a few different ways. For example, you can write tags manually, using an `ods text` statement. But, SAS has also provided a built-in avenue for embedding audio and video starting in SAS9.4M3 using the Report Writing Interface (SASd and SASe). The code below creates an HTML file that includes both audio and video content using the Report Writing Interface (RWI):

```
ods html5
  path=filedir
  file="example_audio.html"
  headtext="<style>.usertext {text-align: center;}</style>";

%let base_url = https://upload.wikimedia.org/wikipedia/commons/;
%let video_file_name = Waves_on_the_Thames.webm;
%let audio_file_name = 1-George_Frideric_Handel_-
_Water_Music_Suite_in_F_major_%28Overture%29_HWV348.ogg;

data _null_;
dcl odsout obj();

obj.format_text(data: "Waves on the Thames", style_elem: "SystemTitle");
obj.format_text(data: "Enjoy this water-themed video that shows off
HTML5.");
obj.video(
  file:
"&base_url.transcoded/9/98/&video_file_name./&video_file_name..480p.webm",
  type: "webm",
  width: "854",
  height: "480",
  preload: "auto",
  autoplay: "yes",
  poster: "&base_url.thumb/9/98/&video_file_name./800px--
&video_file_name..jpg",
  loop: "on"
);


obj.format_text(data: "Handel's Water Music Suite in F major (Overture)",
style_elem: "SystemTitle");
obj.format_text(data: "And here is some more water-themed media that shows
off HTML5's features!");
obj.audio(
  file: "&base_url.6/6e/&audio_file_name.",
  type: "ogg",
  preload: "auto",
  autoplay: "off",
```

```
  loop: "no"
);
obj.delete();
run;
ods text="<span id='attribution'>
  Video courtesy of
  <a href='https://commons.wikimedia.org/wiki/User:Secretlondon'>
    secretlondon
  </a>
  [<a href='https://creativecommons.org/licenses/by-sa/4.0/deed.en'>
    CC BY-SA 4.0
  </a>],
  <a href='https://commons.wikimedia.org/wiki/File:&video_file_name.'>
    via Wikimedia Commons
  </a>
  <br>
  Music by George Frideric Handel
  [<a href='https://creativecommons.org/licenses/by-sa/3.0'>
    CC BY-SA 3.0
  </a>],
  <a href='https://commons.wikimedia.org/wiki/File%3A&audio_file_name.'>
    via Wikimedia Commons
  </a>
</span>";
ods html5 close;
```

The `obj.video()` method is used to embed the video within the document. The types of video files that are allowed to be passed through are MP4, OGG, and WEBM. It is important to know that different browsers support different video types. For example, Internet Explorer only supports MP4, where Google Chrome supports MP4, OGG, and WEBM. The safest video type is MP4 because this is supported by almost every single web browser. The `poster` option gives you a placeholder image until the video starts playing (either by the user pressing play or through autoplay).

Similar to embedding video, you can use the `audio()` method. Of course there are different parameters to each method. Check the RECOMMENDED READING section for links to learn more.

As mentioned above, audio and video can be embedded as well using a simple `ods text` statement. This drops the raw HTML right into your document within a *paragraph* node. It is a great alternative to using the RWI if you have not already upgraded to SAS 9.4M3. For example, the code below drops a video into your report:

```
ods text="<video width='854' height='480' controls>
  <source
    src='https://upload.wikimedia.org/wikipedia/commons/
         transcoded/9/98/Waves_on_the_Thames.webm/
         Waves_on_the_Thames.webm.480p.webm' type='video/webm'>
</video>";
```

The content of media you embed is completely up to you. In some cases, you could add an audio file of yourself explaining content in the document and in other cases you could add uplifting music for the recipient to listen to as he/she goes through your findings. With ODS HTML5 these options are endless.

CUSTOM TABLES OF CONTENTS

A table of contents in your document can greatly increase its usability, especially if the report contains a lot of output. The idea is that you have links at the beginning of the document that take you directly to a desired section further down in the document. This helps your audience navigate the document and get the information that's important to them.

Custom tables of contents aren't new to SAS ODS reports. Previously there were two main ways to create custom tables of contents: 1) through the use of framesets or 2) through the use of custom anchors. With the release of HTML5, framesets are now obsolete since they damage the usability and accessibility of an HTML document. Now, only the latter method is available in ODS HTML5. That's OK because using HTML with CSS (and even JavaScript) gives us the ability to do way more than we could with framesets alone. By interweaving CSS and JavaScript into your reports, you can make great looking documents that are just as slick as anything you see on the web.

Let's take a look at how to implement a very simple custom table of contents using PROC ODSTEXT. This ties together almost everything we've learned so far in this paper:

```
/* BEGIN WITH ODS HTML5 STATMENT */
ods html5
  /* SPECIFY FILE PATH AND FILE NAME */
  path=filedir
  file="example_toc.html"

  /* SPECIFY DOCUMENT TITLE (IN HEADER) */
  (title="Fish Catch Data by Species")

  /* NO MACHINE-GENERATED TOC */
  options(outline="no")

  /* SPECIFY STYLING FOR TABLE OF CONTENTS */
  headtext="
    <style>
    #toc ul {
      width: 200px;
      margin: auto;
      border: 1px solid #c1c1c1;
      padding: 15px 0px 5px 30px;
    }
    </style>";

/* DOCUMENT HEADING */
ods text="<h1>Finland's Lake Laengelmavesi Fish Catch Data By
Species</h1>";

/* GET ALL DISTINCT SPECIES AND PUT THEM IN A MACRO VARIABLE */
proc sql noprint;
  select distinct species
  into :species_list separated by " "
  from sashelp.fish
  order by 1;
quit;

/* DEFINE A MACRO TO POPULATE TABLE OF CONTENTS WITH SPECIES IN DATASET */
%macro create_toc_list(species_list);
  %do i=1 %to %sysfunc(countw(&species_list.));
    item "<a href='#Species&i'>%scan(&species_list.,&i.)</a>";
  %end;
%mend create_toc_list;

/* CREATE A UNIQUE ID FOR THE TABLE OF CONTENTS */
ods anchor="toc";

/* USE ODSTEXT PROCEDURE TO MAKE TABLE OF CONTENTS */
proc odstext;
```

```
  p "Table of Contents" /
  /* DEMONSTRATING INLINE STYLING */
    style={
      htmlstyle="
        text-align: center;
        font-weight: bold;
        font-size: 1.2em;
        color: #112277;
    "};
  list; %create_toc_list(&species_list.);
run;

/* GIVE EACH BY-SECTION IN PRINT PROCEDURE A UNIQUE ID TO LINK TO */
ods anchor="Species1";

/* SORT DATA AND PRINT RESULT */
proc sort data=sashelp.fish out=fish;
  by species;
run;

proc print
  data=fish
  label
  style(table)={posthtml="<a href='#toc'>Back to Table of Contents</a>"};
  by species;
run;

/* CLOSE ODS HTML5 DESTINATION */
ods html5 close;
```

CONCLUSION

SAS gives you many different options for outputting your findings. You can generate reports in rich text format (*.rtf*), portable document format (*.pdf*), hypertext markup language (HTML), and many more. One of the more popular output formats in SAS is HTML. Recently, a new version of HTML was released called HTML5. With this change to HTML, SAS decided to follow suit and release a new output destination known as ODS HTML5. One of the biggest new additions with ODS HTML5 is that the default graphics device is now in the scalable vector graphics (*.svg*) format. There are numerous advantages to using a SVG format for all of your graphics in SAS compared to the previous default graphics device (PNG).

In addition to this new default graphics format, SAS also introduced many new topics in ODS HTML5 to improve your capabilities in SAS. A couple of the more interesting additions that we covered were the ability to embed audio and video files directly in your output and the ability to create animated/interactive charts to include in your output. These new ideas can add tremendous usability to your output files.

We also touched on some topics that were already rolled out in SAS but may help make your usage of ODS HTML5 even better. The use of cascading style sheets (CSS) is an important topic in HTML5. It allows you to control look and feel of your reports. Using the power of CSS, you can make reports look and behave exactly how you want them to. Another key point that we touched on was the ability to control plots. You can determine how you want your graphics to be outputted and even tell SAS to embed your graphics in an HTML file or output them as a stand-alone graphics file. One final idea we touched on was the use of a custom table of contents. Including a table of contents in your output will greatly increase the usability and accessibility of the document.

Your SAS output files are how you display the fruits of your work to your audience. Often, no one will see the code or know the hours or work that you put into the analysis. Your ODS report will be the face of the

analysis for your clients. For that reason, making the most of your output files is extremely important in any line of work. So, do yourself a favor and *Take a Dive into HTML5*. Come on in, the water's fine!

REFERENCES

Mincey, Marvin. "Have you created Scalable Vector Graphics with SAS?". 9 December 2014. Available at < https://blogs.sas.com/content/sgf/2014/12/19/have-you-created-scalable-vector-graphics-with-sas/>

SAS. "Cascading Style Sheets and ODS." Available at <https://support.sas.com/rnd/base/ods/templateFAQ/Template_csstyle.html>

SAS. "Create an Animated Graph." 16 November 2017. Available at <http://documentation.sas.com/?docsetId=grstatug&docsetTarget=p1xw4cbegqxrshn184m0n13yjfky.htm&docsetVersion=9.4&locale=en>

SAS. "Creating Accessible Output with the Output Delivery System." 8 November 2017. Available at <http://documentation.sas.com/?docsetId=odsug&docsetTarget=p0i7wdj6chqccin1uwndkz6ezf10.htm&docsetVersion=9.4&locale=en>

SAS. "Example 2: Adding Audio to HTML5 Output." 8 November 2017. Available at <http://documentation.sas.com/?docsetId=odsadvug&docsetTarget=p0re9domly2xf3n1n1pj2w14fstg.htm&docsetVersion=9.4&locale=en>

SAS. "Example 3: Adding a Video to HTML5 Output." 8 November 2017. Available at <http://documentation.sas.com/?docsetId=odsadvug&docsetTarget=p14nsf2n9a8ui2n0z938ms0sbumc.htm&docsetVersion=9.4&locale=en>

SAS. "What's New in the Output Delivery System." 8 November 2017. Available at <http://documentation.sas.com/?docsetId=odsug&docsetTarget=odsugwhatsnew94.htm&docsetVersion=9.4&locale=en>

W3C. "HTML5 Differences from HTML4." W3C Working Group Note 9 December 2014. 2014. Available at <http://www.w3.org/TR/2014/NOTE-html5-diff-20141209/>.

W3C. "Mathematical Markup Language (MathML) Version 2.0 (Second Edition)." W3C Recommendation 21 October 2003. 2003. Available at <http://www.w3.org/TR/2003/REC-MathML2-20031021/>.

W3C. "Scalable Vector Graphics (SVG) 1.1 (Second Edition)." W3C Recommendation 16 August 2011. 2011. Available at <http://www.w3.org/TR/2011/REC-SVG11-20110816/>.

ACKNOWLEDGMENTS

The authors would like to thank Tashi Reigle, Kyle Bedell, Dr. Nick Andersen, and Dr. Nick Duesbery for their editorial input.

We would like to thank Vecteezy.com for providing the vector image of the diving SAS programmer.

We would also like to thank Wikimedia.org for providing the video and audio file for the section on embedding video and audio in ODS HMTL5.

RECOMMENDED READING

- *ODS HTML5 Statement*
- *What's New in the Output Delivery System*
- *Creating Accessible Output with the Output Delivery System*
- *SAS® 9 ODS Cascading Style Sheet (CSS) Tip Sheet*
- *W3C Cascading Style Sheets Home Page*

CONTACT INFORMATION

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

Brandon George
georgeb@mail.gvsu.edu