An introduction to MathML, SVG and JavaScript

by

Edward Ellis (Department of Mathematical Sciences, Brunel University, email: apgege@brunel.ac.uk)

Introduction

This article covers some elementary HTML, MathML, SVG. It also covers creating dynamic documents of all three using JavaScript. This originally came to my attention when I worked for Dr. Greenhow on Mathletics. Mathletics is a CAA system that uses HTML, MathML, SVG and JavaScript to produce dynamic questions. For instance, a student may be presented with a problem involving moments where the weights and distances are chosen randomly. Using MathML and SVG, all equations and diagrams can adapt to depict valid equations/diagrams for whichever random values are chosen. Although that work was focused on CAA, it is really applicable to almost any web based system. People have asked how these things are done. Hopefully this article will serve as an adequate introduction.

Not all browsers support either MathML or SVG natively. MathML examples in this page are either provided in HTML, or are images rendered by Design Sciences WebEQ product. All SVG diagrams in this page are cropped screen shots from Internet Explorer with the Adobe SVG plug-in. The Dirac Sourceforge page has a fairly comprehensive breakdown of support for these standards with various browsers.

Revision of Hyper-text Markup Language (HTML)

HTML, MathML and SVG are all markup languages. Consequently, they have simmilar aspects. Before we readers look at new markup languages mentioned, we should revise the most common and perhaps more familiar one. This revision is also intended to give readers new to HTML a grasp of the basics.

Getting started

For those new to HTML, you can create an HTML document from notepad. All you have to do is to make sure it contains the following code:

```html
<html>
<head>
<body>
Your HTML code goes here.
</body>
</html>
```

Those of you using windows may also have to change the windows extension to either .htm or .html.
Simple tags

Markup languages are based around tags. Tags delimit areas to be 'marked up'. They usually come in pairs: one to point out the start of the area and the other to point out the end. This is most easily recognised by example.

<table>
<thead>
<tr>
<th>Tag</th>
<th>Start tag</th>
<th>Meaning</th>
<th>Example</th>
<th>Example's HTML</th>
</tr>
</thead>
<tbody>
<tr>
<td>b</td>
<td>&lt;b&gt;</td>
<td>Bold</td>
<td>A word is <strong>bold</strong>.</td>
<td>A word is &lt;b&gt;bold&lt;/b&gt;.</td>
</tr>
<tr>
<td>i</td>
<td>&lt;i&gt;</td>
<td>Italic</td>
<td>A phrase is <em>italic</em>.</td>
<td>A phrase &lt;i&gt;is italic&lt;/i&gt;.</td>
</tr>
<tr>
<td>u</td>
<td>&lt;u&gt;</td>
<td>Underline</td>
<td>This sentence is underlined.</td>
<td>&lt;u&gt;This sentence is underlined&lt;/u&gt;.</td>
</tr>
</tbody>
</table>

Table 1 – Simple tags

Nesting simple tags

Tags may also surround other tags. When this happens, it is important to close the inner most tags before closing the outer most tags.

<table>
<thead>
<tr>
<th>Example</th>
<th>Example's HTML</th>
</tr>
</thead>
<tbody>
<tr>
<td>All words are bold and one is also italic.</td>
<td>&lt;b&gt;All words are bold and one is also&lt;i&gt;italic&lt;/i&gt;.&lt;/b&gt;</td>
</tr>
</tbody>
</table>

Table 2 – Example of nesting simple tags

Nesting more complex tags (tables)

There are also tags that must be used in combination. Tables are a good example. Simple HTML tables can be created from four tags, as shown below.

<table>
<thead>
<tr>
<th>Tag</th>
<th>Start tag</th>
<th>End tag</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table</td>
<td>&lt;table&gt;</td>
<td>&lt;/table&gt;</td>
<td>table</td>
</tr>
<tr>
<td>tr</td>
<td>&lt;tr&gt;</td>
<td>&lt;/tr&gt;</td>
<td>table row</td>
</tr>
<tr>
<td>th</td>
<td>&lt;th&gt;</td>
<td>&lt;/th&gt;</td>
<td>Table header</td>
</tr>
<tr>
<td>td</td>
<td>&lt;td&gt;</td>
<td>&lt;/td&gt;</td>
<td>Table cell</td>
</tr>
</tbody>
</table>

Table 3 – HTML tags for tables

Tables can include table row tags (tr). Table row tags can only contain table header (th) or table cell (td) tags. Table header tags (th) are used to indicate that this cell is the title or header for all table cell (td) tags in the same row/column. The following example should help to clarify this:

<table>
<thead>
<tr>
<th>Example</th>
<th>Example HTML</th>
</tr>
</thead>
</table>
| First name | Last name               | <table>  
| Chris       | Sangwin                |   <tr> 
| Martin      | Greenhow               |   <th>First name</th><th>Last name</th> 
| Edward      | Ellis                  |   </tr>  
|             |                        |   <tr> 
|             |                        |   <td>Chris</td><td>Sangwin</td> 
|             |                        |   </tr>  |
Table 4 – HTML Table example

**Elementary JavaScript with HTML**

Web pages written purely in HTML are always static. It can be useful to include dynamic content in web pages. The random parameters mentioned in the Introduction are one example of dynamic content. There are a number of methods for providing dynamic content. We will look at JavaScript. JavaScript comes with two different dialects. One dialect is interpreted by the web server, the other by the web browser. We will introduce you to the second dialect.

JavaScript code is easy to include in HTML documents by using the script tags. Let us revisit the example given in table 4, but this time we will use JavaScript. Here the 'Interpreted HTML' is the HTML that the browser interprets the web page as including after it has worked out the JavaScript code.

<table>
<thead>
<tr>
<th>Example</th>
<th>Example HTML</th>
</tr>
</thead>
<tbody>
<tr>
<td>First name</td>
<td>Last name</td>
</tr>
<tr>
<td>Chris</td>
<td>Sangwin</td>
</tr>
<tr>
<td>Martin</td>
<td>Greenhow</td>
</tr>
<tr>
<td>Edward</td>
<td>Ellis</td>
</tr>
</tbody>
</table>

```
<html>
<head>
<script>
function table_of_arrays(header,cells) {
    var start_tbl = '<table border="2">';
    var head_tbl = '<tr>,'
    var cell_tbl = ';
    var end_tbl = '</table>';
    var cols = header.length;
    var rows = cells.length/header.length;
    for (var i = 0; i < cols; i++) head_tbl += 'th>"+header[i]+"</th>
    head_tbl += '</tr>
    for (var i = 0; i < rows; i++)
    cell_tbl += '<tr>,'
    for (var j = 0; j < cols; j++) cell_tbl += 'td>+cells[+j+i]+"</td> cell_tbl += '</tr>
    return start_tbl+head_tbl+cell_tbl+end_tbl;
}
</script>
</head>
<body>
<script>
var titles = new Array("First name","Last name");
var info = new
```
Table 5 – Example table using JavaScript

This shows you some of the simple aspects of JavaScript. Below we list a summary of the most notable features of the example.

1. JavaScript is always surrounded by script tags.
2. Script tags surrounded by head tags can be used to define functions (like `table_of_arrays()`).
3. Script tags surrounded by body tags can call function defined in the head section (like `table_of_arrays()`).
4. The function generates an HTML table from a pair of arrays.
5. This HTML is then copied from the `table_html` variable and placed in the document for the browser to render.
6. JavaScript is interpreted while the browser is deciding how to render the page.

It is important to realise that the HTML table is just text as far as JavaScript is concerned. This allows JavaScript to generate as much HTML as you like, containing any information that you have access to.

For more info on the HTML 4 specification goto: [http://www.w3.org/TR/REC-html40/cover.html](http://www.w3.org/TR/REC-html40/cover.html).

**Elementary Mathematics Markup Language (MathML)**

It is also worth noting that there are what you might like to think of as three different dialects of MathML. Whilst this is not exactly true, it will do to be getting on with. Presentation MathML details how an equation should appear on the screen, but not what it means. Content MathML details what an equation means but does not describe how it should be displayed. Interface MathML is the third dialect and is also beyond the scope of this article. We will focus on Presentation MathML as we are most interested in using MathML to display mathematics.

**Getting started**

Just as with HTML, there is some MathML that every MathML statement includes (like the `html`, `head` and `body` tags).

```xml
<math xmlns="http://www.w3.org/1998/Math/MathML">
... your MathML code goes here ...
</math>
```

**Simple tags**

Just like the simple HTML tags shown before, MathML also has some simple tags you will need to know. There is a significant difference between the simple HTML and simple MathML tags.
The simple HTML tags could surround other tags. This is not the case in MathML, as we shall see.

<table>
<thead>
<tr>
<th>Tag</th>
<th>Start tag</th>
<th>End tag</th>
<th>Meaning</th>
<th>Example</th>
<th>Example code</th>
<th>Example meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>mn</td>
<td>&lt;mn&gt;</td>
<td>&lt;/mn&gt;</td>
<td>(Math) Number</td>
<td>9</td>
<td>&lt;mn&gt;9&lt;/mn&gt;</td>
<td>Number 9</td>
</tr>
<tr>
<td>mo</td>
<td>&lt;mo&gt;</td>
<td>&lt;/mo&gt;</td>
<td>(Math) Operator</td>
<td>=</td>
<td>&lt;mo&gt;=&lt;/mo&gt;</td>
<td>Equals operator</td>
</tr>
<tr>
<td>mi</td>
<td>&lt;mi&gt;</td>
<td>&lt;/mi&gt;</td>
<td>(Math) Identifier</td>
<td>x</td>
<td>&lt;mi&gt;x&lt;/mi&gt;</td>
<td>Identifier named x (identifiers are used for variables and functions)</td>
</tr>
</tbody>
</table>

| Table 6 – Some simple MathML tags |

This allows you to produce MathML expressions for simple equations. For example:

<table>
<thead>
<tr>
<th>Example</th>
<th>Example's Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>x=a+2</td>
<td>&lt;mi&gt;x&lt;/mi&gt;&lt;mo&gt;=&lt;/mo&gt;&lt;mi&gt;a&lt;/mi&gt;&lt;mo&gt;+&lt;/mo&gt;&lt;mn&gt;2&lt;/mn&gt;</td>
</tr>
</tbody>
</table>

| Table 7 – MathML expression example |

There are MathML tags that can include other MathML tags, just as we saw with the HTML table. MathML is more exact in this regard. Many MathML tags can only contain a specific number of tags. This is because the tags that are included are treated like arguments in to a function. Some of these tags are listed below, along with the number or arguments they take and some examples.

<table>
<thead>
<tr>
<th>Tag</th>
<th>Start tag</th>
<th>End tag</th>
<th>Number of elements to include</th>
<th>Meaning</th>
<th>Example</th>
<th>Example's code</th>
</tr>
</thead>
<tbody>
<tr>
<td>msub</td>
<td>&lt;msub&gt;</td>
<td>&lt;/msub&gt;</td>
<td>2</td>
<td>One MathML element with another in subscript.</td>
<td>x_i</td>
<td>&lt;msub&gt; &lt;mi&gt;x&lt;/mi&gt; &lt;mi&gt;i&lt;/mi&gt; &lt;mi&gt;i&lt;/mi&gt; &lt;msub&gt;</td>
</tr>
<tr>
<td>msup</td>
<td>&lt;msup&gt;</td>
<td>&lt;/msup&gt;</td>
<td>2</td>
<td>One MathML element with another in superscript.</td>
<td>2^n</td>
<td>&lt;msup&gt; &lt;mn&gt;2&lt;/mn&gt; &lt;mn&gt;2&lt;/mn&gt; &lt;mn&gt;2&lt;/mn&gt; &lt;msup&gt;</td>
</tr>
<tr>
<td>msubsup</td>
<td>&lt;msubsup&gt;</td>
<td>&lt;/msubsup&gt;</td>
<td>3</td>
<td>One MathML element with the second element in subscript and the third element in superscript.</td>
<td>x_i^2</td>
<td>&lt;msubsup&gt; &lt;mi&gt;x&lt;/mi&gt; &lt;mi&gt;i&lt;/mi&gt; &lt;mn&gt;2&lt;/mn&gt; &lt;mn&gt;2&lt;/mn&gt; &lt;msubsup&gt;</td>
</tr>
<tr>
<td>mfrac</td>
<td>&lt;mfrac&gt;</td>
<td>&lt;/mfrac&gt;</td>
<td>2</td>
<td>A fraction, the first element in</td>
<td>a/b</td>
<td>&lt;mfrac&gt; &lt;mi&gt;a&lt;/mi&gt; &lt;mi&gt;b&lt;/mi&gt; &lt;mi&gt;b&lt;/mi&gt;</td>
</tr>
</tbody>
</table>


the numerator and the second in the denominator.

Table 8 – MathML tag examples

The last tag we will cover in this section allows you to group MathML expressions into a single MathML element. This is demonstrated below.

<table>
<thead>
<tr>
<th>Example</th>
<th>Example's Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>$2^{x+3}$</td>
<td><code>&lt;msup&gt;&lt;mn&gt;2&lt;/mn&gt;&lt;mrow&gt;&lt;mi&gt;x&lt;/mi&gt;&lt;mo&gt;+&lt;/mo&gt;&lt;mn&gt;3&lt;/mn&gt;&lt;/mrow&gt;&lt;/msup&gt;</code></td>
</tr>
</tbody>
</table>

Table 9 – MathML example with expression as superscript

Here, as you can see, we used the mrow tag to group the elements in the superscript together. This was necessary to ensure that the msup tag directly contained only two MathML elements.

Earlier we saw how to use JavaScript to insert HTML code in a web page. You can also use JavaScript to include MathML in a document in exactly the same fashion. For example, the Mathletics library includes a function displaypolynomial() (which can be seen at http://www.brunel.ac.uk/~mapgege/displaypolynomial.script) that generates Presentation MathML for a polynomial from an array. It works in a similar fashion to the table_of_arrays() function we covered in the HTML section. This article does not include an explanation, but a live demo of MathML, SVG and JavaScript can be found at: http://www.brunel.ac.uk/~mapgege/sinerule.html.

There is far more to MathML than we have seen so far. The complete specification can be found at http://www.w3.org/TR/MathML2/.

Elementary Scalable Vector Graphics (SVG)

Getting started
Like HTML and MathML, there is some code that you must include in any SVG document to start with. This is shown below.

```xml
<svg width="4cm" height="4cm" version="1.1" xmlns="http://www.w3.org/2000/svg">
Your SVG code goes here.
</svg>
```

Clearly you should overwrite the given values of the height and width with your own values.
Simple tags

One of the major differences between HTML and MathML was that MathML included some elements that could only contain a set number of elements. MathML communicated a lot of information by describing which elements contained which other elements. SVG by contrast communicates information by setting arguments inside the open tag that starts an element. The table below lists some of these elements.

<table>
<thead>
<tr>
<th>Tag</th>
<th>Tag</th>
<th>Meaning</th>
<th>Example</th>
<th>Example's code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Line</td>
<td>&lt;line&gt;</td>
<td>Straight line from point (x1,y1) to (x2,y2).</td>
<td><img src="example.png" alt="Line_example" /></td>
<td>&lt;line x1=&quot;50&quot; y1=&quot;5&quot; x2=&quot;50&quot; y2=&quot;95&quot; stroke=&quot;blue&quot; stroke-width=&quot;10&quot; /&gt;</td>
</tr>
<tr>
<td>Circle</td>
<td>&lt;circle&gt;</td>
<td>Circle centred at point (cx,cy) with radius r.</td>
<td><img src="example.png" alt="Circle_example" /></td>
<td>&lt;circle cx=&quot;50&quot; cy=&quot;50&quot; r=&quot;50&quot; /&gt;</td>
</tr>
<tr>
<td>Rectangle</td>
<td>&lt;rect&gt;</td>
<td>Rectangle with top, left point at (x,y) with given height and width.</td>
<td><img src="example.png" alt="Rectangle_example" /></td>
<td>&lt;rect x=&quot;15&quot; y=&quot;25&quot; height=&quot;50&quot; width=&quot;70&quot; stroke=&quot;green&quot; fill=&quot;yellow&quot; stroke-width=&quot;5&quot; /&gt;</td>
</tr>
</tbody>
</table>

Table 10 – SVG examples

You also saw, in the examples above, the attributes "stroke", "stroke-width" and "fill". These define the colour of the line (for line) or border (for shapes), the width of the previous, and the colour with which the inside of the shape should be shaded. All visible tags in HTML, MathML and SVG have attributes that can change their size, colour and other display related characteristics. These can be used to enhance the accessability of a web-page.

The importance of order

Like MathML, SVG places importance in the order in which elements are described. The first element described is 'drawn' first with each successive element being drawn in order. This can easily be seen by the following examples.

<table>
<thead>
<tr>
<th>Example</th>
<th>Example code</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="example.png" alt="Example" /></td>
<td>&lt;svg height=&quot;100&quot; width=&quot;100&quot; viewBox=&quot;0 0 100 100&quot;&gt; &lt;circle cx=&quot;50&quot; cy=&quot;50&quot; r=&quot;25&quot; stroke=&quot;red&quot; fill=&quot;white&quot; stroke-width=&quot;5&quot; fill-opacity=&quot;0&quot;/&gt; &lt;line x1=&quot;10&quot; y1=&quot;10&quot; x2=&quot;90&quot; y2=&quot;90&quot; stroke=&quot;blue&quot; stroke-width=&quot;5&quot; /&gt; &lt;line x1=&quot;10&quot; y1=&quot;90&quot; x2=&quot;90&quot; y2=&quot;10&quot; stroke=&quot;green&quot; stroke-width=&quot;5&quot; /&gt; &lt;/svg&gt;</td>
</tr>
</tbody>
</table>
Using JavaScript with HTML and SVG

Earlier we saw how to use JavaScript to insert HTML code in a web page. You can also use JavaScript to include SVG in a document in exactly the same fashion. However, since the page will include both HTML and SVG, you will need to give the browser a helping hand so it can work out which one is which! One example is shown in Table 12.

<table>
<thead>
<tr>
<th>Example</th>
<th>Examples code</th>
<th>Interpreted HTML and SVG</th>
</tr>
</thead>
</table>
| Below is a labeled angle. | <?xml version="1.0"?>
<html xmlns:svg="http://www.w3.org/2000/svg" xmlns="http://www.w3.org/1999/xhtml">
<head>
<object id="AdobeSVG" classid="clsid:78156a80-c6a1-4bbf-8e6a-3cd390eeb4e2"></object>
<?import namespace="svg" implementation="#AdobeSVG"?>
<title>SVG Demo.</title>
<script>
function ns_shaded_sector(origin,theta1,theta2,r,label,stroke,stroke_width,fill,ns) {
    var scale = 0.6;
    var a = new Array( Math.cos(theta1) * radius , -Math.sin(theta1) * radius);
    var b = new Array( Math.cos(theta2) * radius , -Math.sin(theta2) * radius);
    var a_to_b = new Array( b[0] - a[0] , b[1] - a[1] );
    var p_a_to_b = new Array( a_to_b[0] * scale , a_to_b[1] * scale );
    var mid_pt = new Array( ( a[0] + b[0] ) / 2 , ( a[1] + b[1] ) / 2 );
    var path_code = 'M '+origin[0] +','+origin[1] +
    ' l '+a[0]*scale +','+a[1]*scale +
    ' a '+r +','+r +
    ' 0 0 0 ' +label +
    ' a '+r +','+r +
    ' 0 0 0 ' +label +
    ' l '+b[0]*scale +','+b[1]*scale +
    ' z';
    path_code += 'l '+a[0]*scale +','+a[1]*scale +
    ' a '+r +','+r +
    ' 0 0 0 ' +label +
    ' a '+r +','+r +
    ' 0 0 0 ' +label +
    ' l '+b[0]*scale +','+b[1]*scale +
    ' z';
}
</script>
</head>
<body>
<p>Below is a labeled angle.</p>
<svg:svg width="100px" height="100px"
viewBox="0 0 100 100">
<path d="M 10,70 l 48,0 a 80,80 0,0 -`
</svg>
path_code += 'a '+r+',r' 0 0,0
  +p_a_to_b[0]+';+p_a_to_b[1]+' z';
var line_code = 'c<+ns+:line
  x1="'+origin[0]+'" y1="'+origin[1]+'"
  x2="'+(origin[0]+a[0])+'"
  y2="'+(origin[1]+a[1])+'"
stroke="'+stroke+'" stroke-
  width="'+stroke_width+'"'/>
;line_code += '<'+ns+':line
  x1="'+origin[0]+'" y1="'+origin[1]+'"
  x2="'+(origin[0]+b[0])+'"
  y2="'+(origin[1]+b[1])+'"
stroke="'+stroke+'" stroke-
  width="'+stroke_width+'"'/>
;line_code += '<'+ns+':text
  x="'+(origin[0]+(mid_pt[0]*0.4))+'"
  y="'+(origin[1]+(mid_pt[1]*0.4))+'"
  font-
  size="16pt"
fill="'+stroke+'">a</'+ns+':text>

} </script> 
</head> 
<body> 
<p>Below is a labeled angle.</p>

<script>
var radius = 80; 
var origin = new Array(10,70); 
var stroke = "blue";
var stroke_width = 2; 
var fill = "yellow";
var svg_code = '<svg:svg width="100px" height="100px" viewBox="0 0 100 100">';
svg_code += ns_shaded_sector(origin,0/Math.PI/4,radius,"a",stroke,stroke_width,fill,"svg");
svg_code += '</svg:svg>'; 
document.write(svg_code); 
</script> 
</body> 
</html> 

Table 12 – Example of using JavaScript to insert HTML and SVG code
Table 12 deserves some explanation. The reader should note that the red box around the labelled angle was inserted manually to highlight the borders of the SVG image.

Some new code has been added. The most significant of these are the xml tag (the first line) and the arguments in the html tag. On the first line, the xml tag tells the browser that the document is XML. This is necessary to allow us to use XML namespaces to combine both SVG and XHTML (very similar to HTML) in the same page. The arguments in the html element create the namespaces. First, the xmlns:svg argument tells the browser that all elements with the prefix svg: are SVG elements. You will notice that throughout this document all svg elements have the prefix 'svg:'. For example the '<svg:svg ... >' tag. Afterwards, the second argument, xmlns tells the browser that the default namespace is XHTML.

Later, the object tag and the import tag are for Internet Explorer. The object tag tells it that it will need the Adobe SVG viewer plugin, while the import tag tells the plugin need only worry about the elements in the svg: namespace. You will recall that in the introduction you read that different browsers had different levels of support for MathML and SVG. If you want this page to be visible in browsers other than Internet Explorer (with SVG plugin) you will want to edit this code a little.

There are other ways of integrating HTML and SVG, neither of which have been covered in this article. You can use the embed tag, or you can insert a link to an SVG document in your HTML page using the anchor (a) element. However, scripting in these case is more complex.

For those that are interested, the complete specification of SVG can be found at http://www.w3.org/TR/SVG/index.html. SVG is not limited to the simple images used in this article. An example of a more complex image can be found at http://www.levien.com/svg/tiger.svg. Readers that would like a real world example of the code above should look at http://www.brunel.ac.uk/~mapgege/pie_chart.html. You can also use JavaScript to make your web pages interactive. That is beyond the scope of this article, but those that are interested will find Adobe have some excellent demonstrations at http://www.adobe.com/svg/examples.html.

Acknowledgment
I should like to thank Dr. Martin Greenhow of Brunel University for his useful discussions while writing this article.