Wrangling Math from Microsoft Word into JATS XML Workflows

A guide to working with math in Word

Wrangling Math from Microsoft Word into JATS XML Workflows

Wrangling Math from MS Word into JATS XML

Caitlin Gebhard1 and Bruce Rosenblum2\*

1 Customer Success Department, Inera Incorporated, 19 Flett Road, Belmont, MA 02478

2 Management Department, Inera Incorporated, 19 Flett Road, Belmont, MA 02478

Correspondence: Caitlin Gebhard (info@inera.com)

† B Rosenblum presented this paper at JATS-Con 2016

Teaser

This paper clarifies the different forms of equations that can be encountered in Word documents and discusses the issues and idiosyncrasies of converting these various forms to MathML, LaTeX, and/or images in the JATS XML model.

Abstract

Mathematics is a fundamental building block of modern technology, research, and industry, and yet the technological means of publishing mathematics can still be surprisingly challenging. As a result, anyone involved in producing, publishing, or reading mathematical equations electronically knows that writing and publishing math is not a simple process.

The majority of today’s scholarly papers are authored in Microsoft Word. Some of those papers include simple and/or complex math. Authors have multiple means at their disposal to insert equations in Word documents, including several of Word's native equation editors and third-party applications, such as Design Science’s MathType. Building workflows that smoothly and accurately transform all of these formats into the appropriate XML markup for use in multiple rendering environments has many challenges.

This paper clarifies the different forms of equations that can be encountered in Word documents and discusses the issues and idiosyncrasies of converting these various forms to MathML, LaTeX, and/or images in the JATS XML model. It also touches on workflow alternatives for handling equations in various rendering environments and how those downstream requirements may affect the means of equation extraction from Word documents.

**Keywords**: Math, Microsoft Word, XML, JATS

**Abbreviations**: JATS, Journal Article Tag Suite

Received: March 1, 2016; Revised March 14, 2016; Accepted: March 21, 2016

Introduction

Mathematics is a fundamental building block of modern technology, research, and industry, and yet the technological means of publishing mathematics can still be surprisingly challenging. As a result, anyone involved in producing, publishing, or reading mathematical equations electronically knows that writing and publishing math is not a simple process. The Internet allows us to share text and images seamlessly across platforms and systems, but similar demands for mathematical equations can fall short. We expect equations to look great and be useful (e.g., searchable, readable to the visually impaired, and perhaps even manipulable or solvable) on any electronic delivery system. In short, published mathematics need to be both human readable *and* machine readable.

Published math should discoverable, accessible, and sustainable; treating equations simply as graphics does not suffice. In other words, the benefits of an XML publishing workflow should extend to equations. JATS supports display and inline equations that “can be expressed as ASCII characters, as a graphic, or using TeX, LaTeX, or MathML mathematics expressions” [1]. Each of these supported formats comes with its own benefits—and baggage—and can be created from a Microsoft Word document.

Authors have various ways of adding equations to their Word documents, including several Word-native equation editors and third-party applications, such as Design Science’s MathType. The focus of this paper is to explain the various ways that authors may insert equations into Word documents and how to build workflows that smoothly and accurately transform all of these formats into the appropriate XML markup. While this paper focuses specifically on JATS, the concepts discussed apply to any Word-to-XML workflow regardless of whether the XML model is JATS, DITA, DocBook, or TEI.

Before we dive in, we acknowledge those who say that there are no good solutions for publishing math content using Microsoft Word. TeX and LaTeX are excellent solutions for STM content, and this paper touches lightly on the history and benefits of those tools. However, note that this paper is *not* about TeX and LaTeX; our audience is specifically those who must handle equations in Word documents. For those interested in learning more about LaTeX-XML workflows, we recommend a study on River Valley’s workflow [2].

A little bit of history

In the days of type-script manuscripts, it was common for equations to be drawn-in by hand; for example, Figure 1 shows a 1914 manuscript prepared by Albert Einstein.

**Figure 1.** Typed manuscript by Albert Einstein and Marcel Grossmann, published in 1914 in the *Zeitschrift für Mathematik und Physik* [3].

Typesetting mathematics from such manuscripts in the days of hot metal was a time-consuming manual task. The introduction of photocomposition systems by Photon Corporation in the 1950s attracted the attention of those concerned with scientific communication. In 1953co-founder of the Graphic Arts Research Foundation Vannevar Bush[[1]](#footnote-2) wrote to Charles Dollard of the Carnegie Foundation regarding a “photographic paging machine” that could (and did) improve the quality and complexity of publication production. Bush proposed that “a composing machine for mathematical publications could be made. This would be a real boon to scientific publication, and I would like to see it done. This has some real problems involved, but there are good approaches” [5]. However, photocomposition did not solve the problems associated with mathematical publications. Rendering mathematical characters while preserving the spacing and alignment of complex equations presented ongoing challenges.

TeX and LaTeX

In the 1970s Donald Knuth, professor emeritus at Stanford University, created the typesetting programming language TeX[[2]](#footnote-3) in frustration about the quality of the typesetting of his classic work *The Art of Computer Programming*. TeX is a standard language for author-generated mathematical typesetting. It is important to note that TeX is not XML or SGML, and it is mainly concerned with document layout and formatting rather than document structure. Several variations of TeX now exist, including LaTeX, which is commonly used in scientific and technical publishing. TeX is also used by many page composition firms to typeset a wide range of content, especially that which is math-intensive.

Designed with complex mathematics in mind, TeX allows for fine-tuned control over the layout of a manuscript, including kerning and placement of mathematical functions. TeX became very popular very quickly, due largely to the rise of information technology in the 1970s and 1980s; almost every mathematician and scientist had newly discovered access to a PC, workstation, or microcomputer [6].

However, the flexibility afforded by TeX comes with a price; the language is quite complicated. In 1985 the computer scientist and Turing Award winner Leslie Lamport create LaTeX (pronounced “lah-tech” or “lay-tech”), a powerful document preparation system that allows the user to more easily and consistently employ the flexibility and precision of TeX [7].[[3]](#footnote-4) Using either a GUI (i.e., WYSIWYG) editor or a simple text editor, authors write a LaTeX input file that contains text and commands for processing and formatting that text. The author then compiles the input file using the LaTeX program, which produces a device-independent file (DVI) that can be used to generate PDF or PostScript files. LaTeX boasts an expansive library of fonts, symbols, and functions, which often makes it a first-choice authoring tool for mathematicians, physicists, and others who produce and typeset complex equations [8]. Now maintained by the LaTeX3 Project, LaTeX is used primarily to typeset complex documents such as journal articles, technical reports, and books in the scientific and technical fields.

Modern word processors

TeX provides an excellent solution for writing, editing, and typesetting complex documents. However, outside of mathematics, physics, and a few other math-intensive disciplines, most scholarly authors have gradually shifted from hand-written or typed manuscripts to commercial word processing software other than TeX or LaTeX.

Most of this transition occurred between 1985 and 2000, driven by the dramatic drop in costs (in real dollar terms; i.e., accounting for inflation and increased computing performance) of personal computers, the advent of easy-to-learn and easy-to-use graphical user interfaces, and the growth of commercial word processing applications such as WordPerfect and Microsoft Word. Most nontechnical or non-tech-savvy authors of scholarly content found these applications far easier to learn than TeX. These commercial applications also provided greater availability to formal IT support as well as to informal colleague support than TeX. Expansion of Word, in particular, into scholarly authoring was driven by Microsoft’s bundling of Office software, commercial Word add-ins for citation and reference management,[[4]](#footnote-5) and support for the more complex aspects of research papers: special characters, tables, and images. While TeX and other commercial word processors have all of these features, Microsoft came to dominate the market. By 2000 most papers submitted to scholarly journals were in Word format.

Microsoft Word also included the ability to add equations. However, if there was only one way to add equations to a Word file, there would not be a need for this paper. In fact, by 2007 six different methods of adding equations to Word existed. We have found that many editorial and production teams, especially those in organizations that do not work with large amounts of mathematics, are not familiar with these variants and the unique handling that each needs in a publication workflow. The core of this paper aims to remedy this by describing each of these methods and the issues of integrating them into an XML publishing workflow.

Enter MathML

TeX and LaTeX solved the problem of creating *graphical representations* of mathematical content. Generating *better* graphical representations, however, is not necessarily the solution to the myriad complexities of publishing digital math content.

The American Association of Publishers (AAP) Math model, the original foundation of SGML math markup for journal publication, emerged in the 1990s. The math model of an important early SGML markup scheme, ISO 12083: Electronic Manuscript Preparation and Markup, although not directly derived from the AAP DTD, was developed and somewhat based on a review of the AAP DTD [9].

However, both LaTeX and the AAP/12083 XML model for math suffered limitations when looking beyond presentation-only math. “There was a danger during the 1990s that a standard would emerge for mathematical representation on the web that would be based on a TeX- or ISO 12083-like typesetting language. This would have been disastrous because it would have precluded, or made far more difficult, meaningful computational interaction with mathematical expressions found on the web” [10]. Partly as a solution to this dilemma, MathML was designed to go beyond presentation-only math and facilitate the use and re-use of math content across multiple technological platforms. In the late 1990s, starting at a conference sponsored by Wolfram Research, MathML was developed into a W3C recommendation [11].

MathML is an application of XML for describing the visual and semantic meaning of an equation. MathML consists of a number of XML tags that can be used to mark up equations in terms of either their presentation (i.e., the formatted appearance of the equation when displayed) or their content (i.e., the semantics of the equation). Presentation MathML focuses on the display of the equation whereas Content MathML focuses on the meaning of expression, and its elements represent the functions applied in a formula. MathML is a very rich language, but similar to TeX and LaTeX, it is a computer representation and not meant to be written by hand. Instead, programs such as MathType transform WYSIWYG equations into MathML.

By 2002, when development started on JATS, those scholarly publishers who used XML were divided among TeX, MathML, and AAP/12083 math. The original NLM DTD working group realized that it was essential to support TeX and MathML for archive and publishing purposes, and hooks were left in the DTD to accommodate AAP/12083 math. However, MathML took off quickly as the preferred XML representation of math shortly after NLM DTD 1.0 was released in 2003,[[5]](#footnote-6) and so the AAP/12083 model was never added.

Benefits of using MathML

Many benefits exist to representing math content with MathML instead of using purely presentational methods such as graphics. First, MathML is a key player in the development of “math aware” search engines. Keywords for math content are often over-general; as the late Robert Miner (former VP of Research and Development at Design Science) describes, “One can search for ‘quadratic polynomial’, but there is no effective way to narrow the search to a particular polynomial or class of polynomials” [12]. What’s more, different fields of study can use very different terminology for identical math objects. Instead, imagine typing a math equation—or part of a math equation—into a search engine and getting back a list of publications in which it occurs! While a MathML-based search engine is not yet commercially available, several promising academic prototypes are in development.[[6]](#footnote-7)

In addition to *discoverability*, MathML provides the additional advantage of *accessibility*. To meet federal and state accessibility laws, images must be supplemented with alt text, but text descriptions of math equations do not adequately address the needs of their users. Despite efforts to standardize, the written—or spoken—descriptions of math equations can vary from publication to publication. With MathML, math content can be accessed by users with a variety of accessibility needs. Systems such as MathJax, Design Science’s MathPlayer, and the MathSpeak Initiative use complex heuristics to convert MathML into a variety of output formats for large print, Braille, and audio [16-18].[[7]](#footnote-8)

MathML also helps publications reach greater *sustainability*. XML-centric publishing workflows help keep articles usable over time, even as the methods of rendering that XML evolve. Because math literature tends to stay relevant for a very long time, an XML workflow is ideal for making that content usable throughout its lifespan.

MathML offers a lot of great benefits to publishers of math content, and because it is a standard for representing mathematical formulae, MathML can be created and re-used across many different platforms, including common editing tools such as LaTeX and Microsoft Word.

Getting math into and out of Word

Math published today in JATS is represented as one (or more, using the <alternatives> element) of the following formats, depending on the requirements of the publisher:

• Text (with Unicode characters and enhanced with simple font face changes)

• Graphics

• MathML

• TeX

However, math in Word does not necessarily start in these formats. Math can be entered in Word using the following methods:

• Text (with Unicode characters and enhanced with simple font face changes)

• Graphics

• Formula fields

• Design Science MathType

• Microsoft Equation Edit 3.0

• Microsoft Equation Builder

Each of these methods is described in more detail in the following sections, including the methods, issues, and challenges of converting each format to JATS XML.

Math as text

Simple mathematical formulae (i.e., those expressions that can be expressed on a single line) can be typed into Word using the keyboard. Letters, numbers, and basic math symbols can be typed on an ASCII keyboard, and font face changes such as italic, bold, superscript, and subscript can be applied with Word character formatting. This method has been available since Word was first developed in the mid-1980s. Most special (non-ASCII) symbols can be typed as Unicode values or entered via Word’s “Insert Symbol” function if a font containing the required symbol is available.

Special symbols

Entering mathematical “special symbols” could be challenging in Word’s early years. Unicode did not exist when the earliest versions of Word were developed. Word included a Symbol font with a useful but limited math set, and many special needs were filled with a wide range of custom math symbol fonts that were each limited to 232 characters. Individual characters were addressed by font name and character offset within the font. This lack of standardization for special symbol fonts caused more than a few woes when converting Word files to XML, especially when more creative authors made custom fonts for themselves with the specific characters they needed. The STIX font project was developed in part to alleviate the problems caused by large numbers of special non-Unicode compliant math fonts [19].

More recent versions of Word have full Unicode support and much more extensive support for math symbols in newer fonts such as Calibri and Cambria. Today, most mathematical symbols and other special symbols can be inserted from Word’s Insert Symbol menu. Alternatively, if you know the four-digit Plane 0 Unicode (hexadecimal) value of a character, you can type the value of a character, select the value, and use the ALT+X keyboard shortcut to convert the character value to the corresponding character glyph (e.g., typing “03B2” and pressing ALT+X inserts a lowercase Greek beta in the current font).

Converting text equations to XML

Text equations can be transformed from Word DOCX format to JATS XML. These keyboarded equations can be treated as plain text; for example, the expression

*x* ∈ [10,350]

can be simply transformed to

<italic>x</italic> &#x2208; [10,350]

in JATS. Both <disp-formula> and <inline-formula> allow Unicode text and font face changes, so this could also appear as

<inline-formula><italic>x</italic> &#x2208; [10,350]</inline-formula>

or, correspondingly,

<disp-formula><italic>x</italic> &#x2208; [10,350]</disp-formula>

if it is a display formula. Text conversions such as these can be done by extracting the document.xml fragment of a DOCX file and then applying an XSLT transformation to create JATS XML.

Structured math required

Some publishers require that all math be tagged as MathML or TeX, even inline equations that can be typed at the keyboard.[[8]](#footnote-9) This presents a challenge in conversion, because such publishers must define exactly what constitutes inline math. For example, *n* < 5 is probably inline math; however, is *n*? Once a set of rules has been developed for what constitutes an inline equation, those equations must be converted to either MathML or TeX. This can be done by rekeying typed math using one of Word’s math editors and converting as described in following sections. Alternatively, the text can be converted to JATS XML, and then the equations can be converted to MathML or TeX from the XML. This process can be partially automated, but some equations will likely require manual attention.

Converting poorly typed math

When converting keyboarded inline math to XML, typically some manual cleanup is required to overcome authors who are either lazy or creative typers. Consider the character ± (plus-minus, U+00B1). It is all too often entered as an underlined plus symbol, +. Once, we saw a paper written by a scientist who knew a minus sign was different from a hyphen but could not figure out how to insert the symbol, so he typed an underscore character and applied superscript formatting!

Improperly typed math may look similar to its semantically correct counterpart, but the difference between plus-minus and an underlined plus is important when the characters are published in XML. When the XML is then converted to text by text-to-speech or text-to-braille accessibility devices, the resulting equation is described inaccurately or nonsensically.

Furthermore, while it is not likely that complex equations would be entered as manually inserted ASCII characters, it is important to note that some formatting tricks in Word (e.g., font size, tabs, combined superscript/raised formatting) can be lost when keyboarded equations are converted to XML unless great care is taken in proofing.

Graphic images

From the very beginning, Word supported the insertion of images into Word files. Some authors create equations as images, typically using separate image-editing applications, and then copy-and-paste those images into Word. While not common, this method is a source of frustration for those tasked with converting equation images to XML because there is no automated way to convert the images to XML. Usually, the only way to convert them is for someone to carefully rekey the images into one of the math editors described in the next few sections.[[9]](#footnote-10)

Another option is simply to leave those equations as graphics. However, the graphic format itself is often problematic. Devices for consuming content vary widely in capability and resolution, and most equation graphics found in Word files are bitmaps that are not scalable like vector graphics. An article published in *Research Information* recounts several of the issues surrounding graphical representations of math: “Images do not display well at large zoom, do not print at full resolution, and do not consistently align well with the surrounding text. This is particularly problematic on tablets and other mobile viewing devices with small screens” [20]. These problems only exist in nonvector graphic formats.

Beyond the issues with displaying math graphics, images of math content do not take advantage of the layers of information that digital publishing can afford. Images are not machine-readable and they do not represent the semantics of the equation. Unless the equation is provided with an alt text description, equations handled as graphics are essentially lost to readers who access the math content via a screen reader or other accessibility device.

Formula fields

Fields in Word are used to create and manipulate variables in the document, such as page numbers or mail merge labels. Fields may also be used to perform calculations or to display mathematical formulae.

Expression and Formula fields existed in Word for Windows as early as Word version 1.x. In Word versions 6.x and later, the Expression and Formula fields were renamed to Formula and Equation, respectively [21].

Formula and Equation fields are somewhat difficult to add to Word documents, and as a result, they have never been widely used. However, production teams may occasionally encounter papers that have these fields.

Entering Equation fields

By the time Word 2007 rolled around, Word included three equation editors (described in the following sections), and so the feature to add Equation fields was well hidden in the Word user interface. Adding an Equation field in Word 2010 requires six clicks just to reach the Equation field editor.[[10]](#footnote-11)

Detecting Equation fields

All fields in Word can be made easier to detect by turning on Field Shading.[[11]](#footnote-12) This feature in Word applies a gray shade to all fields, making it much easier to see when they have been used in a document.

As you scroll through the document and see equations with a gray wash, they may be an Equation field, a Word Equation Edit 3.0 object, or a MathType Equation. To determine if the equation is an Equation field, select the equation and right click. If the context menu offers an “Edit Field” option, then it is an Equation field (Figure 2).

**Figure 2**. The “Edit Field” option appears on the context menu for an Equation field.

Converting Equation fields

Microsoft does not provide any direct way to extract Equation fields. Images of equation fields can be created by saving a Word document to HTML, but these images are low resolution bitmaps and not vector graphics suitable for scaling on a device of any resolution.

Also, no direct way exists to convert Equation fields to XML. It is possible to write an application to parse Microsoft’s field format. However, an easier method is to use MathType’s Convert Equation feature, which can convert these fields to MathType objects. Once the conversion is complete, the equations can be exported to scalable EPS graphics or converted to MathML or TeX using MathType’s native functions.

Design Science MathType

MathType is a full-featured equation editor originally developed by Design Science in 1987. It can be used with Microsoft Office, Adobe InDesign, Wordpress, and many more applications as an OLE-compatible plugin. MathType provides a WYSIWYG editor for manipulating equations within the document as well as tools for global equation formatting.

MathType offers a variety of translators for exporting equations in a document to different formats. In addition to conversions to image files (EPS or GIF), MathType 6.9 includes translators for TeX and LaTeX, including custom extensions for publishers (e.g., AMSTeX and AMSLaTeX for the American Mathematical Society). MathType also includes several different formats of MathML for use in different XML workflows: MathML 1.0 and MathML 2.0 with m namespace, namespace attribute, or no namespace. As indicated on their website, Design Science continues to create new translators, and publishers are able to customize or create their own translators using the MathType Software Development Kit [22]. The MathType SDK is available to download on the Design Science website free of charge, although Design Science does not offer technical support for this tool [23].

The MathType SDK also includes documentation on how to access the MathType API, which provides methods to automate the conversion of MathType equations to MathML, TeX, or graphics. The MathType API can thus be used to maximize automated conversions of Word to XML in publishing workflows.

Microsoft Equation Editor

In 1991 Microsoft licensed a subset of MathType from Design Science and dubbed it “Microsoft Equation Editor.” It first appeared in Word for Windows 2.0. Equation Editor, a simplified version of MathType,[[12]](#footnote-13) is a formula editor that (similar to MathType) allows users to create and edit equations within the WYSIWYG environment of Word.

As a subset of MathType, Equation Editor has many of the same functions and operations of MathType. Its toolbar includes quick access to both common and complex mathematical symbols as well as different formatting options. Once an equation is inserted into the Word document, the math symbols or formulae can be managed as a Word object.

Equation Editor was positively received; as one user wrote to *PC Magazine* in 1992, “People whose writing involves numeric equations will find the Equation Editor in *Word for Windows* 2.0 well-suited to its task—much more friendly and more capable than the equation fields in *WinWord* 1.*x*.” One complaint from the same user is that Equation Editor is hidden; Word had no built-in command to quickly access the Equation Editor interface. He writes, “Using the Equation Editor would be even easier if there were a way to assign one of the Toolbar buttons to an ‘Insert Equation’ command” [25].

The macro to create such a button, as suggested by *PC Magazine* writer M. David Stone, alludes to the interesting relationship between Word and Equation Editor. Stone writes, “Because the Equation Editor is a separate application, the *WinWord* macro recorder ignores everything you do while in the Equation Editor” [25]. While you can create a macro to open the Equation Editor tool, you cannot create a macro to insert a particular equation using that tool.

Getting from MathType and Microsoft Equation Editor to XML

Microsoft Equation Editor

Microsoft Equation Editor provides a good platform for adding equations to Word documents. However, it does not provide a useful way to extract those equations from Word. While you can save your document to HTML and collect the (non-scalable) bitmap images, that’s about as far as you can go. To go further, you’ll need MathType.

MathType to XML

One of MathType’s great benefits is its ability to import and export equations in different formats, including LaTeX and MathML. Equations in either markup language can be imported into a document using copy and paste or drag and drop, and then edited and formatted as a MathType object. In the reverse direction, equations can also be copied from Word to the clipboard either in plain-text MathML (i.e., as you would in an XML or text editor) or in MathML clipboard format [22].

From a production standpoint, MathType provides a full set of options to either convert equations to TeX or MathML or export those equations to EPS or GIF graphics. Additionally, the conversions to MathML, TeX, or any other text-based (i.e., not graphic) format can be customized, and new conversions can be developed from scratch. MathType conversions are controlled by TDL files, and the format for TDL files is documented in the MathType SDK [23]. These features often make MathType the tool of choice for publishers and vendors that work with equation objects in Word-to-XML publishing workflows.

Microsoft Equation Editor to MathType to XML

Because Microsoft Equation Editor is a subset of MathType, MathType can import Equation Editor math into MathType, and then MathType can be used to convert those equations for XML workflows.

Users who have licensed MathType will find that equations created using Equation Editor are automatically “upgraded” to MathType objects. However, it is important to note that Equation Editor objects (those that have not been converted to MathType objects) are not identical to MathType objects. Design Science explains on their FAQ: “There have been no significant changes to Equation Editor since we licensed it to Microsoft in 1991. MathType, on the other hand, has been continually upgraded and improved” [26].

What does this mean? Equations that were created with Equation Editor are automatically accessible and editable to those with MathType installed, but MathType objects cannot be edited with Equation Editor alone. In the MathType for Windows Help dialog, Design Science suggests that users with Equation Editor download the thirty-day free trial of MathType.

Its functionality is not limited in any way. After the trial period has ended, MathType runs in what we call Lite mode. This provides similar functionality to Equation Editor. However, with MathType Lite you or a collaborator can edit all equations created with MathType 6, making it the ideal tool for people who need to collaborate with you on technical documents. [27]

MathType’s Convert Equations feature

Although not common, there are occasional errors in equations that are converted from MathType to MathML. Usually these errors arise because of how the author has created the equation, especially if the author had used a much older version of MathType. The best way to resolve these problems is to use MathType’s Convert Equations function, taking care to select “MathType Equations (OLE Objects)” as the target type (Figure 3).

**Figure 3**. MathType Convert Equations dialog.

This convert function can clean up and rebuild equations that were created with older versions of MathType (select the MathType and Equation Editor option on the left side of the dialog). MathType can also be used to convert other math objects—including MS Word EQ formula fields and Equation Builder OMML equations—to MathType objects. The consistency afforded by this option can be invaluable.

A word of caution: Experience among publishers has found that one should reproof equations converted from OMML or Equation fields to MathType before proceeding with publication to ensure that no errors have occurred during the conversion. Equations converted from Microsoft Equation Editor and earlier versions of MathType have a lower probability of conversion problems because the underlying formats were designed by Design Science.

MathType to XML issues

MathType for Microsoft has a couple of known issues that can affect your publishing workflow, although to the credit of Design Science, many past issues have been fixed in more recent versions of MathType. Known issues include the following.

• Authors can create equations (typically with superscript or subscript constructs) that do not convert to MathML correctly. While most of these cases convert to MathML correctly in MathType 6.5 and later, once in a while an equation will need to be manually rekeyed.

• Some symbols may not be rendered correctly when the equation is saved to EPS. In most cases, this problem can be resolved by following the instructions in MathType TechNote 114 [28], but there are some less common cases (e.g., euro symbol, accented Greek letters) that require special attention.

Despite a couple of issues, MathType is one of the more popular Word equation editor tools in the STM community because it has been available since Word manuscript submissions first became common, and because it can export to MathML, TeX, and graphics.

MathType to graphics

As noted, MathType equations can be exported to EPS or GIF graphics. While these are not XML formats, they can be useful in publication workflows that do not support native rendering of MathML or TeX equations. For example, InDesign does not have native support for MathML or TeX.[[13]](#footnote-14) In an XML workflow with InDesign, some publishers export an equation as both EPS (for InDesign rendering) and XML or TeX (for web rendering). In such cases, it is best to use the JATS <alternatives> element, as shown in this example from the JATS tag library [30]:

<disp-formula id="pbio-0020328-e001">

<alternatives>

<mml:math display="block" xmlns:mml="http://www.w3.org/1998/Math/MathML">

<mml:mrow><mml:msub>

<mml:mrow><mml:mtext>Strength</mml:mtext></mml:mrow>

<mml:mi>i</mml:mi></mml:msub>

<mml:mo>=</mml:mo>

<mml:mi>log</mml:mi>

<mml:mo>&ApplyFunction;</mml:mo>

<mml:mfrac>...</mml:mfrac>

<mml:mo>=</mml:mo>

...

</mml:math>

<graphic xlink:href="pbio.0020328.e001.gif"/>

</alternatives>

</disp-formula>

Word Equation Builder

Microsoft introduced a brand new equation editor in Word 2007 as a replacement for Microsoft Equation 3.0. The new equation editor was dubbed “Equation Builder.” It was designed for tighter integration into the Word document environment and for more elegant math typography than was previously available in Word. It is located on the Insert ribbon, under the Equation icon.[[14]](#footnote-15)

Equation Builder natively uses XML markup called Office Math Markup Language (OMML), a mathematical markup language that is part of the XML file formats introduced with Microsoft Office. OMML is based on Unicode Technical Report 25, *Unicode Support for Mathematics* [31].

As described in *Creating Research and Scientific Documents Using Microsoft Word*, “In the past, the default equation editor that shipped with early versions of Word had a negative reputation, for its inconsistent functionality and less-than-attractive output. In striking contrast, the equation editor built in to Word 2013 is a powerful and flexible tool” [32]. OMML allows a much wider range of formatting features to be used with Word equations than Equation Editor or MathType, including footnotes, Word comments, color text, and Track Changes. Word’s Find and Replace can even be used on Equation Builder objects.[[15]](#footnote-16)

A new math font was developed alongside Equation Builder: Cambria Math, part of the Cambria font family. Murray Sargent, a Partner Software Design Engineer at Microsoft, describes the benefits: “High-quality low-resolution screen display is very important for the way people work with documents in the Internet age: most documents are perused on screen and only printed for purposes of detailed examination. This is a major advantage of our math system” [33]. The enhanced typographical capabilities of this tool can make it a good choice for those who publish PDFs directly from Word.

STM publishing community backlash

However, most scholarly publishers do not publish PDFs directly from Microsoft Word. Instead, they import Word files into applications such as InDesign, or they convert Word files to XML and then typeset the XML with applications such as InDesign, 3B2, or XPP.

The new DOCX file format—and especially the new Equation Builder feature—proved disruptive to the scholarly publishing community when Office 2007 was released, most notably because Equation Builder math was not backwards compatible with previous versions of Word.

Word 2007 was a radical change from earlier versions because of the new user interface, new XML-based file format, and new equation editor. Because the changes were so large, many publishers waited as long as two or even three years to move to Word 2007 (or directly to Word 2010) from Word 2003 (or earlier). While Microsoft did provide a free compatibility package that allowed Word 2003 to read DOCX files, the OMML equations appeared as fuzzy (i.e., poor quality) graphics in Word 2003. The result was similar when DOCX files were saved to DOC format from Word 2007, and publishers found that those fuzzy graphics could not be converted to XML unless they adopted both Word 2007 and DOCX file format.[[16]](#footnote-17)

As a result, some publishers, such as *Nature* and *Science*, declined to accept manuscripts submitted in the then-new DOCX format for a number of years until they upgraded to DOCX-compatible versions of Word. In June 2007 the second author addressed these problems in an open letter to Microsoft, detailing the STM publishing community’s concerns with Word 2007. He explained:

For most scholarly publishers, the challenge is to publish high-quality and accurate information on a regular schedule. Software upgrades to critical publishing systems, unless they are seamless or provide a significant immediate benefit, are often not a priority.

In the case of Word 2007, upgrading is not seamless. Because files incorporating OMML equations are not semantically backwards compatible with older versions of Word, publishers must update an entire ecology of systems before they can accept DOCX files. Completing such updates requires work with third parties, careful testing, training, and finally deployment—often one system at a time—of updated applications. All of this takes time.

In the meantime, because a DOCX file with OMML equations renders the equations as graphics when used with today’s [Word 2003-based] systems, it’s easier for publishers to ask authors to refrain from submitting DOCX files until every part of the workflow ecology is DOCX-compatible. And not just updated to accept DOCX, but also updated so that OMML can seamlessly be integrated into systems today that provide publishers with full-text XML and tagged math according to the NLM DTD or other 12083-derived DTDs.

The full letter can be read on *Nature*’s archived blog, *Nascent* [34].[[17]](#footnote-18)

Although Equation Builder’s initial reception by most of the STM publishing community was less than enthusiastic, recently it has been accepted by many publishers as they have updated to newer versions of Office and conceded that authors will ultimately submit math using whatever tool seems to be most handy. That stated, many publishers do have their preferences. For example, *PLoS One*’s author instructions suggest, “We recommend using MathType for display and inline equations, as it will provide the most reliable outcome. If this is not possible, Equation Editor is acceptable” [36], and *Science Magazine*’s instructions say: “*Science* prefers to receive files in Word’s .docx format; however, we advise against creating math equations using Word 2007’s equation editor. Please instead format equations in Mathtype [sic] or using the legacy equation editor in Word” [37].

OMML to MathML

Word 2007 shipped with the capability to convert OMML to MathML. However, the feature was well hidden and poorly documented by Microsoft when Word 2007 shipped. It was also buggy. The conversion provided in the original release of Word 2007 would sometimes create MathML that did not validate against the MathML DTD. While Microsoft gradually made fixes in service packs and Word 2010, the transforms were not stable until Word 2010.

The transform from OMML to MathML (and also MathML to OMML) in Word 2007 (and later) is performed by two XSLT scripts that ship with Word: omml2mml.xsl and mml2omml.xsl. These scripts can be used outside of Word if you are reading or manipulating DOCX XML files directly. These scripts enable users to copy and paste equations via MathML, although this use of the Word clipboard is turned off by default. To turn on this support in Word 2007, you must open the Design ribbon by selecting or creating an Equation Builder equation and click the down arrow to the right of “Tools” to open the Equation Options dialog menu; then, you must select the option “Copy MathML to the Clipboard as plain text” [38].

It is helpful to note that “the MathML clipboard formats are only available if the selected text is completely contained within a math zone,” as Sargent explains. “The MathML formats cannot represent text in a math zone along with text not in a math zone. You need to use a format like RTF or HTML to copy such combinations” [38]. While clipboard support is useful for working with individual or a small number of equations, it is impractical if you want to process a whole document.

You can also use the math language transforms included in Word to convert all equations at once. To do so, save the Word document as an HTML file, and then transform the HTML to XHTML or XML.[[18]](#footnote-19) The equations in the XHTML file will require some cleanup; David Carlisle of the Numerical Algorithms Group has supplied a xhtml-mathml transform for this purpose. Then, you can apply the Microsoft supplied omml2mml.xsl stylesheet to the math fragments [40].

Converting between Equation Builder and MathType

As noted, some publishers have distinct preferences for either Equation Builder or MathType. It is possible to convert between these two formats using MathML as the interchange format. As with all such conversions (especially those going through essentially two conversions; i.e., Format A to MathML to Format B), one should be careful to reproof the equations to ensure that no errors or significant format changes have been introduced by the conversion.

Equation Builder to MathType

MathType provides an easy way to convert selected or all Equation Builder equations in a Word document to MathType. Simply select the Convert Equations dialog (as noted previously), select “Word 2007 and later (OMML) equations” on the left and “MathType equations (OLE Objects)” on the right. MathType calls Microsoft’s omml2mml.xsl transform to convert each equation to MathML, and then the resulting MathML is converted into MathType format.

MathType to Equation Builder

Those who prefer Equation Builder to MathType can convert MathType objects to Word Equation Builder objects by converting them to MathML and then to OMML. Dadi Gudmundsson of Sensor Analytics has documented his migration process on Sargent’s blog. In short, you would export the MathType equations to MathML using MathType; use Find/Replace to edit the MathML for compatibility with Word (e.g., Find "<mml:math" and replace with "<mml:math xmlns:mml="http://www.w3.org/1998/Math/MathML"); and then create and use a macro to copy the edited MathML and paste it as an Equation Builder object. Gudmundsson recommends recording the following steps as a macro for converting the MathML to Equation Builder object [41]:

a. Open the Find feature, turn on wildcard & ignore whitespace selections, enter "?mml:math\*/mml:math?" into the Find field and push Find. The first block of Mathml code should now be highlighted.

b. Push Ctrl+X to cut the selected text.

c. Push Alt+= to insert an equation.

d. Select paste special and paste as unformatted text (in majority of cases the mathml code in question will now appear as a human readable equation).

Gudmundsson explains that there are a few issues with this process, but overall the conversion works well for Equation Builder workflows [41].

OMML to anything else

Equation Builder works well if you are creating a PDF file from Word, if you are creating HTML with GIF images, or if you are converting Word files to XML in a workflow for which only MathML is required. However, if you need TeX or scalable graphics, the workflow will be more challenging because Microsoft does not provide methods to convert or save Equation Builder math to any formats besides OMML, MathML, or GIF. One of the reasons many publishers prefer MathType to Equation Builder math, and why many publishers convert all Equation Builder equations to MathType at the start of the production workflow, is because they can easily convert MathType equations to many different formats. Additionally, the MathType SDK and API provide further support for customizing and automating the workflow.

OMML to TeX

Getting TeX out of Word is not a native Microsoft feature. If you have MathType installed, you can convert all equations to MathType objects, and then convert those to one of the many variants of TeX supported by MathType.

Without MathType, several scripts are available online created by editors and authors who have encountered this challenge. One such tool is Pandoc, a free “swiss army knife” for converting one markdown language to another. Pandoc can be used to convert documents with math equations to HTML and MathML, HTML with LaTeX or TeX, LaTeX or TeX documents, Word documents with OMML, and many other formats [42].

OMML to scalable graphics

Microsoft has not provided a native method to save Equation Builder math to any scalable graphic format such as EPS. If you have MathType installed, you can convert all equations to MathType objects, and then export those to EPS files.

Without MathType, you can convert the OMML equations to MathML and then use a third-party rendering tool to convert the MathML to EPS, SVG, or other scalable graphic formats. Third-party options include pMML2SVG, a MathML-to-SVG XSLT [43], and MathJax, whose API can be used to generate customized SVG files from MathML [44].

OMML versus MathML

When Microsoft introduced OMML-based equations, there was a lot of discussion about why a new math XML markup language was needed when MathML already exists as a successful language for representing equations. As Sargent describes: “The main problem is that Word needs to allow users to embed arbitrary span-level material (basically anything you can put into a Word paragraph) in math zones and MathML is geared toward allowing only math in math zones” [45]. OMML is more closely aligned with Word’s internal languages, which allows for better integration with many of Word’s features, such as Track Changes and AutoCorrect.

The MathML and OMML that are created from Word deal exclusively with Presentation MathML. However, there are several key differences between the two languages. OMML uses explicit argument tags, while MathML “determines arguments by position” [45]. For example, in the built-up fraction , the OMML representation explicitly tags the numerator and denominator:

<m:oMath xmlns:m="http://schemas.openxmlformats.org/officeDocument/2006/math">

<m:f>

<m:num>

<m:r><m:t>a</m:t></m:r>

<m:r><m:t>+</m:t></m:r>

<m:r><m:t>b</m:t></m:r>

</m:num>

<m:den>

<m:r><m:t>c</m:t></m:r>

</m:den>

</m:f>

</m:oMath>

In MathML, the same fraction is tagged as:

<mml:math

<mml:mfrac>

<mml:mrow>

<mml:mi>a</mml:mi>

<mml:mo>+</mml:mo>

<mml:mi>b</mml:mi>

</mml:mrow>

<mml:mi>c</mml:mi>

</mml:mfrac>

</mml:math>

Note that the semantic labels “numerator” and “denominator” are not explicit in the previous Presentation MathML.[[19]](#footnote-20) Sargent summarizes the two main differences between OMML and MathML as [45]:

1. MathML built-up objects may be described by an infix notation, while OMML’s are described by a prefix notation

2. MathML built-up object arguments are defined positionally, while OMML’s are tagged explicitly.

Workflow implications and issues

Consistent results

There are many ways to create and edit equations in Microsoft Word, but the best tool and method to create and export those equations for use in JATS XML depends on your workflow needs.

A key factor in determining a workflow is consistency. If you intend to output inline and display equations consistently in the XML, then you probably want to use the same method for editing and creating that output.

There are few technical workflow issues for text inline equations created outside of MathML, TeX, or graphics. However, as noted, there is an obvious editorial consideration: What is inline math, and is it a requirement to tag it with math elements or can it simply be plain text?

MathType and Equation Builder MathML

When you convert equations from Word to MathML, you may encounter differences between the MathML created by different equation editors. MathType objects will convert to a slightly different MathML than Equation Builder objects. For example, the equation



created in MathType and Equation Builder yield the following MathML snippets:

|  |  |
| --- | --- |
| MathType | Equation Builder |
| <mml:math><mml:mrow>**<mml:munder>**<mml:mo>&#x2211;</mml:mo><mml:mrow><mml:mi>i</mml:mi><mml:mo>,</mml:mo><mml:mi>j</mml:mi></mml:mrow>**</mml:munder>**</mml:mrow></mml:math> | <mml:math><mml:mrow>**<mml:msubsup>**<mml:mo>&#x2211;</mml:mo><mml:mrow><mml:mi>i</mml:mi><mml:mo>,</mml:mo><mml:mi>j</mml:mi></mml:mrow>**</mml:msubsup>**</mml:mrow></mml:math> |

While both variations of MathML are correct, including both MathType and Equation Builder objects in your Word document may yield inconsistent MathML in your final XML file. For this reason, you may decide to standardize your workflow by converting all math in Word files to one of either MathType or Equation Builder.

Beyond editorial

While we have the tools to manipulate math on the editorial side, that hard work does not necessarily translate into the end product. We may have come a long way from physically copying and pasting the author’s handwritten equation, but math is often still published as images. Of course, these images come from a variety of editorial origins with varying degrees of quality, but the graphical math object still lacks the semantic meaning that enables discoverability, accessibility, and sustainability.

HTML/browser support

One of the reasons math is still published as graphics is the lack of support for the alternative. MathML 1 was released as a W3C recommendation in April 1998, and it was the first XML language to be recommended by the W3C. This makes MathML comparatively mature as a web technology. It is therefore surprising that native support for MathML on the Internet is patchy at best.

In 2013 Peter Krautzberger, project lead for MathJax, summarized the state of browser support for MathML in the article “MathML Forges On.” At that time, MathML development projects had been started and stopped, or ignored completely, by major Internet browsers. In particular, Microsoft’s Internet Explorer and Google’s Chrome teams refused to support MathML, the latter flatly stating that “MathML is not something that we want at this time” [17].[[20]](#footnote-21)

For updated information about browser support for MathML, the website *Can I Use* publishes a useful table of native MathML support [47]. Although MathML is well integrated in HTML5, most browsers do not support it.

At this time, as native browser support for MathML has wavered, MathJax has gained broader adoption for rendering MathML in browsers. MathJax is an open-source JavaScript engine designed to render mathematics in any web browser and e-reader without any additional setup on the part of the end user or reader. Using web-based fonts already supported by modern browsers, MathJax produces high-quality typesetting of math from input MathML, TeX, LaTeX, or AsciiMath. This text-based math is scalable, searchable, and accessible to screen readers and other accessibility devices.[[21]](#footnote-22) MathJax and the MathJax-node API project can also be used to generate custom SVG files and improve handling of graphical and MathML representations of math in accessibility devices [16,49]. While MathJax is a good solution for many publishers, rendering of tagged math online is still a moving target, and it may continue to be for years to come.

EPUB support

For XML-to-EPUB workflows, the options are also mixed. Created by the International Digital Publishing Forum (IDPF), EPUB 3 supports Presentation MathML within the scope of XHTML content documents. Content MathML can be embedded as well, but only within a special <m:annotation-xml> element. As the specification explains, EPUB supports only the Presentation subset of MathML “to ease the implementation burden on Reading Systems and to promote accessibility, while retaining compatibility with HTML User Agents” [50].

Furthermore, IDPF recommends including an alternative description of mathematical formulae.[[22]](#footnote-23) This fallback is important even for MathML representations of equations, because many screen readers are still unable to read MathML [51]. This alt text can be included as part of the editorial process in Word and subsequently tagged in either <alt-text> or the JATS <long-desc> element.

However, many EPUB reading devices still do not support MathML, so graphics are widely used in conjunction with tagged math. For more information about MathML in EPUBs, we recommend reading Mike Dean’s JATS-Con 2013 presentation, “The Challenges and Benefits of Automating NLM-to-ePub3 File Conversion” [52].

While LaTeX documents can be easily converted into PDFs, additional post-publication file formats such as HTML for the Internet and EPUB for the e-reader require third-party applications, some better than others. For a study of one publisher’s LaTeX-to-EPUB workflow, see River Valley’s workflow study [2]. Note that images were required for complex math formulae that could not be represented using their HTML equivalents.

Conclusion

Incorporating math into XML can be one of the more challenging aspects of an XML workflow. The challenges are complex, because the equations destined for XML can come from many different types of equation editors when working in the Microsoft Word environment. The key to successful integration of all these formats is a clear understanding of each format, of the strengths and weaknesses of each, and of your unique workflow requirements. All of this knowledge, combined with careful training of editorial and production teams on the types of equations and issues they may encounter, can lead to successful and accurate publication of mathematics.

Acknowledgments

We would like to thank Bill Kasdorf, Peter Krautzberger, and Paul Topping for their invaluable feedback during the revision process.

References

<std>1. ANSI/NISO Z39.96-2015. *JATS: Journal Article Tag Suite*, version 1.1.1. Baltimore, MD: National Information Standards Organization, January 2015. Accessed 14 March 2016. http://www.niso.org/apps/group\_public/download.php/15933/z39\_96-2015.pdf</std>

<jrn>2. Rishi, T. “LaTeX to ePub.” *TUGboat* 32, no. 3 (2011): 266–268. Accessed 14 March 2016. https://www.tug.org/TUGboat/tb32-3/tb102rishi.pdf</jrn>

<eref>3. Instituut-Lorentz. Einstein Papers. “Einstein’s 1914 Typescript.” Einstein\_1914\_04. Accessed 14 March 2016. https://www.lorentz.leidenuniv.nl/history/Einstein\_archive/Einstein\_1914\_typescript/Pages/Einstein\_1914\_04.html</eref>

<bok>4. Wiesner, Jerome B. *Vannevar Bush: 1890-1974*. Washington DC, National Academy of Sciences, 1979. Accessed 23 March 2016. http://www.nasonline.org/publications/biographical-memoirs/memoir-pdfs/bush-vannevar.pdf</bok>

<other>5. Vannevar Bush to Charles Dollard [letter], 5 January 1953, private collection.</other>

<jrn>6. Hofmann, Karl H., and Sidney A. Morris. “Editors’ Cut: Managing Scholarly Journals in Mathematics and IT.” *Journal of Research and Practice in Information Technology* 37, no 4 (2005): 299–309. Accessed 14 March 2016. https://www.acs.org.au/\_\_data/assets/pdf\_file/0007/15559/JRPIT37.4.299.pdf</jrn>

<jrn>7. Baramidze, Victoria. “LaTeX for Technical Writing.” *Journal of Technical Science and Technologies* 2, no. 2 (2013): 45–48. Accessed 14 March 2016. http://journal.ibsu.edu.ge/index.php/jtst/article/view/493</jrn>

<jrn>8. Unwalla, Mike. “LaTeX: An Introduction.” *Communicator* (Spring 2006): 33. Accessed 14 March 2016. http://www.techscribe.co.uk/ta/latex-introduction.pdf</jrn>

<conf>9. Rosenblum, Bruce, and Irina Golfman. “A Decade of DTDs and SGML in Scholarly Publishing: What Have We Learned?” Extreme Markup Languages, Montreal, Quebec, 6–9 August 2002. Accessed 14 March 2016. http://conferences.idealliance.org/extreme/html/2002/Rosenblum01/EML2002Rosenblum01.html</conf>

<eref>10. “History of MathML.” *MathML Central*. Wolfram Research. Accessed 14 March 2016. http://www.mathmlcentral.com/history.html</eref>

<eref>11. W3C Math Working Group. “Mathematical Markup Language (MathML).” *W3C Math Home*. Accessed 14 March 2016. http://www.w3.org/Math/whatIsMathML.html</eref>

<jrn>12. Miner, Robert. “The Importance of MathML to Mathematics Communication.” *Notices of the American Mathematical Society* 52, no. 5 (2005): 532–538. Accessed 14 March 2016. http://www.ams.org/notices/200505/fea-miner.pdf</jrn>

<eref>13. “About.” *SearchOnMath*. Accessed 18 March 2016. http://searchonmath.com/about</eref>

<eref>14. KWARC group. *MathWebSearch: Searching Math on the Web*. Jacobs University. Accessed 18 March 2016. http://search.mathweb.org/</eref>

<eref>15. Maths Information Retrieval Research Group. “MiaS.” *MIR@MU*. Masaryk University. Last updated 2 February 2016. Accessed 18 March 2016. https://mir.fi.muni.cz/mias/</eref>

<eref>16. MathJax Consortium. “Accessibility Features.” *MathJax Documentation*. 6 January 2016. Accessed 21 March 2016. http://mathjax.readthedocs.org/en/latest/misc/accessibility-features.html</eref>

<jrn>17. Krautzberger, Peter. “MathML Forges On.” O’Reilly Radar. 1 November 2013. Accessed 14 March 2016. http://radar.oreilly.com/2013/11/mathml-forges-on.html</jrn>

<eref>18. “Learn about MathSpeak.” *MathSpeak*. 2006. Accessed 21 March 2016. http://www.gh-mathspeak.com/learnmathspeak.php</eref>

<eref>19. *STIX Fonts*. Accessed 14 March 2016. http://stixfonts.org/index.html</eref>

<eref>20. Koers, Hylke. “Fulfilling the Potential of Maths Online.” *Research Information* (October/November 2011). Accessed 14 March 2016. http://www.researchinformation.info/features/feature.php?feature\_id=343</eref>

<eref>21. Microsoft Corporation. “WD: Overview of Expression and Formula Field Functions.” *Microsoft Support*. 12 April 2015. Accessed 14 March 2016. https://support.microsoft.com/en-us/kb/105640</eref>

<eref>22. Design Science. “MathType 6.9 Features and Benefits.” *Design Science MathType*. Accessed 14 March 2016. http://www.dessci.com/en/products/MathType/features.htm</eref>

<eref>23. Design Science. “MathType Software Development Kit.” *Design Science MathType*. Accessed 11 March 2016. https://www.dessci.com/en/reference/sdk/</eref>

<eref>24. Design Science. “MathType vs. Equation Editor.” *Design Science MathType*. Accessed 14 March 2016. http://www.dessci.com/en/products/mathtype/mt\_vs\_ee.htm</eref>

<eref>25. Stone, M. David. “The Working Word.” PC Magazine 11, no. 11 (1992): 357–358. Accessed 14 March 2016. https://books.google.com/books?id=WFhT5khImwMC&printsec=frontcover</eref>

<eref>26. Design Science. “MathType FAQ: MathType vs Equation Editor.” *Design Science MathType*. Accessed 14 March 2016. http://www.dessci.com/en/products/mathtype/faqs.htm#mt\_v\_ee</eref>

<other>27. Design Science. “MathType Help: MathType Collaborating with Equation Editor Users.” MathType for Windows, 6.9 [software]. 2010.</other>

<eref>28. Design Science. “TechNote 114: Controlling the Symbol Font Used with Creating EPS Files.” Design Science MathType Support. 10 June 2005. Accessed 14 March 2016. https://mathtype.com/en/support/mathtype/tsn/tsn114.htm</eref>

<eref>29. Schwörer, Ferdinand. “MathTools V2” [movemen brochure]. *movemen GmbH.* October 2014. Accessed 23 March 2016. http://movemen.com/files/downloads/mtv2/MathTools-V2-brochure.pdf</eref>

<eref>30. Journal Publishing Tag Library. NISO JATS Draft Version 1.1d3. National Center for Biotechnology Information (NCBI); National Library of Medicine (NLM), April 2015. Accessed 14 March 2016. http://jats.nlm.nih.gov/publishing/tag-library/1.1d3/</eref>

<eref>31. Beeton, Barbara, Asmus Freytag, and Murray Sargent III. Unicode Support for Mathematics. Unicode Consortium, 31 July 2015. Report no. 25. Revision 14. Accessed 14 March 2016. http://www.unicode.org/reports/tr25/</eref>

<bok>32. Mamishev, Alexander, and Murray Sargent. “How to Work with Equations.” Creating Research and Scientific Documents Using Microsoft Word, chapter 6. Redmond, WA: Microsoft Press, 2013.</bok>

<eref>33. Sargent, Murray. “High-Quality Editing and Display of Mathematical Text in Office 2007.” *Murray Sargent: Math in Office* [blog]. 13 September 2006. Accessed 14 March 2016. http://blogs.msdn.com/b/murrays/archive/2006/09/13/752206.aspx</eref>

<eref>34. Ratner, Howard. “Word 2007 and the STM Publisher Ecosystem.” *Nascent* [blog], Nature.com. 14 June 2007. Accessed 14 March 2016. http://blogs.nature.com/nascent/2007/06/word\_2007\_and\_the\_stm\_publishe.html</eref>

<eref>35. Topping, Paul. “Integrating an Equation Editor into a Document Editor: A Guide for Software Developers” [Design Science white paper]. *Design Science MathType*. March 2011. Accessed 14 March 2016. http://www.dessci.com/en/reference/white\_papers/MathDocEditing/</eref>

<eref>36. “Submission Guidelines.” *PLoS One*. Accessed 8 March 2016. http://journals.plos.org/plosone/s/submission-guidelines</eref>

<eref>37. “Information for Authors: Preparing Your Manuscript and Figures.” *Science/AAAS*. Accessed 8 March 2016. http://www.sciencemag.org/site/feature/contribinfo/prep/</eref>

<eref>38. Sargent, Murray. “MathML on the Windows Clipboard.” *Murray Sargent: Math in Office* [blog]. 27 May 2013. Accessed 14 March 2016. http://blogs.msdn.com/b/murrays/archive/2013/05/28/mathml-on-the-windows-clipboard.aspx</eref>

<eref>39. Cowan, John. “TagSoup: Just Keep on Truckin’.”Accessed 11 March 2016. http://home.ccil.org/~cowan/XML/tagsoup/</eref>

<eref>40. Carlisle, David. “XHTML and MathML from Office 2007.” *David Carlisle* [blog], 10 April 2007. Accessed 14 March 2016. http://dpcarlisle.blogspot.com/2007/04/xhtml-and-mathml-from-office-20007.html</eref>

<eref>41. Sargent, Murray. “Converting Equations from MathType to Word 2007’s Equation Format.” *Murray Sargent: Math in Office* [blog]. 11 February 2007. Accessed 14 March 2016. http://blogs.msdn.com/b/murrays/archive/2007/02/12/converting-equations-from-mathtype-to-word-2007-s-equation-format.aspx</eref>

<eref>42. “About Pandoc.” *Pandoc.org*. Accessed 11 March 2016. http://pandoc.org/index.html</eref>

<eref>43. Piater, Justus, and Alexandre Stevens. pMML2SVG [software]. Accessed 11 March 2016. https://sourceforge.net/projects/pmml2svg/</eref>

<eref>44. MathJax Consortium. “The SVG Output Processor.” *MathJax Documentation*. 6 January 2016. Accessed 21 March 2016. http://docs.mathjax.org/en/latest/options/SVG.html</eref>

<eref>45. Sargent, Murray. “MathML and Ecma Math (OMML).” *Murray Sargent: Math in Office* [blog]. 6 October 2006. Accessed 14 March 2016. http://blogs.msdn.com/b/murrays/archive/2006/10/07/mathml-and-ecma-math-\_2800\_omml\_2900\_-.aspx</eref>

<eref>46. Krautzberger, Peter. “The Curious Invisibility of MathML.” *Peter Krautzberger on the Web* [blog]. 27 October 2015. Accessed 23 March 2016. https://www.peterkrautzberger.org/0184/</eref>

<eref>47. Deveria, Alexis. “Can I Use: MathML.” *CanIUse.com*. Accessed 15 February 2016. http://caniuse.com/#feat=mathml</eref>

<eref>48. Benetech. “MathML Cloud.” *Benetech.com*. 2016. Accessed 11 March 2016. http://benetech.org/our-programs/literacy/born-accessible/mathml-cloud/</eref>

<eref>49. MathJax Consortium. “What Is MathJax?” *MathJax Documentation*. 6 January 2016. Accessed 14 March 2016. http://docs.mathjax.org/en/latest/mathjax.html</eref>

<eref>50. Gylling, Markus, William McCoy, Elika J. Etemad, and Matt Garrish, eds. “2.1.4.1: Embedded MathML.” In *EPUB Content Documents 3.0. Recommended Specification*. International Digital Publishing Forum, 11 October 2011. Accessed 14 March 2016. http://www.idpf.org/epub/30/spec/epub30-contentdocs.html#sec-xhtml-mathml</eref>

<eref>51. Garrish, Matt. “MathML.” In *EPUB 3 Accessibility Guidelines*. International Digital Publishing Forum, 12 October 2015. Accessed 14 March 2016. http://www.idpf.org/accessibility/guidelines/content/mathml/desc.php</eref>

<edb>52. Dean, Mike. “The Challenges and Benefits of Automating NLM-to-ePub3 File Conversion.” In Journal Article Tag Suite Conference (JATS-Con) Proceedings 2015. Journal Article Tag Suite Conference (JATS-Con). Bethesda, MD: National Center for Biotechnology Information (US), 2015. Accessed 14 March 2016. http://www.ncbi.nlm.nih.gov/books/NBK159966/</edb>

Further Reading

<conf>Bahram, Sina, David MacDonald, and CB Averitt. “Enabling Math on the Web, in Word & PDF, Emerging Solutions and Overcoming Issues” [draft conference session submission]. Annual International Technology and Persons with Disabilities Conference (CSUN), 2015. Accessed 14 March 2016. http://davidmacd.com/mathml/making-math-accessible-CSUN-2015L.pdf</conf>

<edb>Bazargan, Kaveh. “A Complete End-to-End Publishing System Based on JATS.” In Journal Article Tag Suite Conference (JATS-Con) Proceedings 2015. Journal Article Tag Suite Conference (JATS-Con). Bethesda, MD: National Center for Biotechnology Information (US), 2015. Accessed 14 March 2016. http://www.ncbi.nlm.nih.gov/books/NBK279828/</edb>

<eref>“Editorial Manager Support for Office 2007 Activated.” Aries Systems press release, 8 April 2008. Accessed 14 March 2016. http://www.ariessys.com/views-press/press-releases/editorial-manager-support-for-office-2007-activated/</eref>

<eref>Inera Inc. “Word 2007 Math” [archived web page]. *Inera.com*. 22 June 2007. Accessed 14 March 2016. https://web.archive.org/web/20090501033911/http://inera.com/word2007math.shtml</eref>

<eref>Inera Inc. “Word 2007 Publishing Update” [archived web page]. *Inera.com*. 7 April 2008. Accessed 14 March 2016. https://web.archive.org/web/20080705015358/http://www.inera.com/word2007update.shtml</eref>

<jrn>Lindeen, Beverly. “Conquering the Production of Mathematical Content.” *Science Editor* 38, no. 2 (2015): 46. Accessed 14 March 2016. http://www.councilscienceeditors.org/wp-content/uploads/v38n2WebPdf.pdf</jrn>

<other>Microsoft Corporation. *Microsoft Word: Word Processing Program for IBM Personal Computer*. Microsoft Corporation, 1983. Accessed 11 March 2016. http://toastytech.com/manuals/MS%20Word%201.00%20for%20DOS%20Manual.pdf</other>

<eref>Parsons, John. “Publishing by the Numbers: The MathML Standard and What It Means for Scholarly and Academic Publishers,” [white paper]. Cenveo Publisher Services, 2016. Accessed 14 March 2016. http://www.intuideas.com/wp-content/uploads/2016/01/MathML\_White\_Paper\_Cenveo\_Publisher\_Services.pdf</eref>

<eref>Sampson, Jonathan. “How Can We Improve the Microsoft Edge Developer Experience? MathML.” *Windows Dev Feedback*. 1 October 2014. Accessed 14 March 2016. https://wpdev.uservoice.com/forums/257854-internet-explorer-platform/suggestions/6508572-mathml</eref>

<eref>Sargent, Murray. “Getting Word 2007 Technical Files into Publisher Pipelines.” *Murray Sargent: Math in Office* [blog]. 13 June 2007. Accessed 14 March 2016. http://blogs.msdn.com/b/murrays/archive/2007/06/13/getting-word-2007-technical-files-into-publisher-pipelines.aspx</eref>

<eref>Sidje, Roger B., and Frédéric Wang. “Mozilla MathML Project.” *Mozilla Developer Network*, 28 June 2015. Accessed 14 March 2016. https://developer.mozilla.org/en-US/docs/Mozilla/MathML\_Project</eref>

<eref>Topping, Paul. “Design Science Proposal to Microsoft to Help STEM (Scientific/Technical/Engineering/Mathematical) Publishers Work with Office 2007 Documents” [Design Science white paper]. *Design Science MathType*. 28 August 2007. Accessed 14 March 2016. http://www.dessci.com/en/reference/white\_papers/STMOffice2007Proposal.htm</eref>

<eref>Topping, Paul. “MathML Workflows in STM Publishing” [Design Science white paper]. *Design Science MathType*. February 2006. Accessed 14 March 2016. https://www.dessci.com/en/reference/white\_papers/mathml\_workflows.htm</eref>

<eref>Wicke, Gabriel. “Introducing Math Rendering 2.0.” MediaWiki list-serv. 23 October 2014. Accessed 14 March 2016. https://lists.wikimedia.org/pipermail/mediawiki-l/2014-October/043482.html</eref>

1. An American engineer, inventor, and science administrator, Bush also headed the U.S. Offce of Scientific Research and Development, acted as president of the Carnegie Institute, and is considered to be one of the greatest influences on the growth of American science and technology in the twentieth century [4]. [↑](#footnote-ref-2)
2. Pronounced “tekh,” it is spelled tau, epsilon, chi and is officially written as <graphic>CGBR16-fa.png</graphic>. In fact, TeX can only be accurately reproduced using TeX. [↑](#footnote-ref-3)
3. It should be pointed out that LaTeX is also “very fussy.” “A trivial mistake may mean that no output is generated and many error messages are displayed” [8]. [↑](#footnote-ref-4)
4. By the late 1980s third parties were hard at work on add-ins to Microsoft Word. One of the earliest add-ins for Word was the citation and reference management software EndNote, and much of the original add-in architecture for Macintosh Word 5 was designed to meet the requirements of EndNote. [↑](#footnote-ref-5)
5. We mean exactly “preferred XML representation of math.” Many publishers still prefer to use TeX math embedded in XML files, and later sections of this paper address converting Word math to TeX as well as MathML. [↑](#footnote-ref-6)
6. Notable prototypes include SearchOnMath, the first search engine to incorporate MathJax [13]; MathWebSearch, developed by the KWARC group at Jacons University [14]; and MIaS (Math Indexer and Searcher), developed by the Maths Information Retrieval Research Group at Masaryk University [15]. [↑](#footnote-ref-7)
7. Presentation MathML is not expressive enough to translate directly into speech in a way that is useful for the end user. As a result, screen readers and other accessibility devices rely on complex algorithms to semantically analyze MathML and then translate it into a more accessible audio representation. This is a complex problem, and the quality of voiced mathematics varies greatly among the available tools [16]. [↑](#footnote-ref-8)
8. The American Physical Society (APS) requires all inline equations be MathML. To reinforce this point, APS’s pre-JATS DTD did not even have superscript and subscript elements, which virtually necessitated conversion of all inline math to MathML. [↑](#footnote-ref-9)
9. “Carefully” is the key word, because rekeyed equations can easily (and often) lack some of the semantics of the original. [↑](#footnote-ref-10)
10. (1) click on the Insert ribbon; (2) click the Quick Parts icon; (3) click Field; (4) select Eq field; (5) click the Field Codes button; and then (6) click the Options button. Now you can start selecting the equation expressions from an alphabetized (non-WYSIWYG) list. [↑](#footnote-ref-11)
11. In Word 2010 and later, click the File ribbon; select Options; select Advanced; under Show Document Content, select *Always* from the Field Shading drop-down menu; and then click OK to save your settings. [↑](#footnote-ref-12)
12. Design Science published a list of differences between MathType and Equation Editor. Equation Editor lacks many of MathType’s features and includes about half the amount of math symbols and templates [24]. [↑](#footnote-ref-13)
13. In recent years, a number of third-party tools have appeared that can be used to render MathML within InDesign, such as movemen GmbH’s MathTools [29]. [↑](#footnote-ref-14)
14. For backwards compatibility, Microsoft Equation 3.0 is still available in current versions of Word; select Microsoft Equation from the Insert ribbon, and then select “Object.” [↑](#footnote-ref-15)
15. However, the “Go To” tool for scrolling through document objects does not support Equation Builder objects, and the “Find and Replace” dialog “Go To” feature, despite listing Equations as an object type, can only find Microsoft Equation Editor and MathType equations, not OMML equations. A workaround when searching for OMML equations is to Find instances of Cambria Math (the default font for Equation Builder objects). [↑](#footnote-ref-16)
16. It should be noted that all was not lost. When you save a DOCX file with OMML equations to DOC, the equations turn to graphics. But if you then resave the DOC file to DOCX format, the equations “rehydrate” into OMML format. Microsoft left enough data behind in the convertor to pull off this magic trick. [↑](#footnote-ref-17)
17. Interesting to note: In 2007 Paul Topping from Design Science proposed in a blog post several new MathType features in response to the STM publishing community’s backlash against Word 2007 (“Design Science Proposal to Microsoft to Help STEM (Scientific/Technical/Engineering/Mathematical) Publishers Work with Office 2007 Documents”). Topping proposed an OMML-to-MathType equation conversion tool, which would “require Microsoft’s assistance, and possibly, bug fixes and small enhancements to Word 2007” [35]. The full proposal can be found here: http://www.dessci.com/en/reference/white\_papers/STMOffice2007Proposal.htm. [↑](#footnote-ref-18)
18. One tool to accomplish this is TagSoup, a SAX-compliant parser that can be used to convert “wild” HTML to clean XHTML [39]. [↑](#footnote-ref-19)
19. While it is not difficult to determine the semantics of this equation, it highlights an overall lack of semantics in Presentation MathML. [↑](#footnote-ref-20)
20. Krautzberger also lamented the lack of visibilty and discussion of MathML on the Internet in his blog post, “The Curious Invisibility of MathML” [46]. [↑](#footnote-ref-21)
21. The MathJax-node project has benefitted tools such as Benetech’s MathML Cloud. Using the MathJax’s SVG output processor and Volker Sorge’s SpeechRuleEngine, Benetech’s MathML Cloud converts MathML to SVG and corresponding math-speech text [48]. [↑](#footnote-ref-22)
22. Although this is recommended by IDPF, implementing a workflow that consistently inserts accessiblity descriptions for equations poses its own set of challenges. [↑](#footnote-ref-23)