A reprint from

American Scientist
the magazine of Sigma Xi, The Scientific Research Society

Writing Math on the Web

Brian Hayes

THE WORLD WIDE WEB was invented at a physics laboratory, and the first users were scientists and engineers. You might think, therefore, that this new channel of communication would be especially well adapted to scientific discourse—that it would facilitate the expression of ideas like

$$\nabla \times \mathbf{E} = -\frac{\partial \mathbf{B}}{\partial t}$$

or

$$\zeta(s) = \sum_{n=1}^{\infty} \frac{1}{n^s} = \prod_{p \text{ prime}} \frac{1}{1 - \frac{1}{p^s}} \cdot$$

If only it were so! The truth is, the basic protocols of the Web offer almost no support for rendering mathematics or other specialized notations such as chemical formulas. Presenting such material on a Web page often requires software add-ons or plug-ins to be installed by the author or the reader or both. Fine-tuning the display of mathematics can be a fussy and finicky process, not much easier than formatting equations with a typewriter. The results sometimes render differently—or not at all—in various Web browsers. This is a sad situation: As the Web has evolved into a thriving marketplace and playground, the scholarly and scientific community that created the technology has not been well served.

The confused state of online mathematical typography is worrisome as well as sad. In years to come the Web will surely be the most important conduit for scientific information. Already it is a major channel for distributing publications and preprints in many disciplines, and it is becoming a venue for less-formal jottings and conversations—everything from homework assignments to blogs. Ideally, the Web would serve as an extension of the blackboard where people gather to talk about science and math during coffee breaks. We need chalk for that blackboard.

The Web would make a dandy blackboard if only we could scribble an equation

The problem is not one of simple neglect. Over the years there have been many earnest efforts to build a reliable facility for writing and reading mathematics online. The trouble is, no one solution has yet gained the kind of widespread adoption that would make it a standard, supported in mainstream Web servers and browsers. Still, there's room for hope. We have technologies that work, if we can agree on how to use them. And a minor change to the infrastructure of the Web might smooth the way for more online math.

Penalty Copy

Even in the world of ink-on-paper publishing, mathematical notation can be a challenge. In the days when printing was done with metal type, equations had to be assembled by hand, piecing together symbols on individual slivers of metal and shimming them into position. A manuscript with a lot of mathematics was known as "penalty copy" because printers would charge extra to compose it.

By the 1970s, metal type was giving way to optical and electronic typesetting devices, which projected letterforms onto photographic film. In principle the new machinery might have aided mathematical typesetting because characters could be placed with equal facility at any position and scaled to any size. But the software for running the phototypesetting machines offered no easy way to exploit this flexibility. The printed product was often inferior to expertly set metal type.

Enter Donald E. Knuth of Stanford University, who was so discontented with the deteriorating quality of mathematical typography that he set aside other work and undertook to build his own typesetting system. The eventual result was TeX, a formatting language not just for equations but for entire documents. Leslie Lamport, now of Microsoft Research, soon introduced LaTeX, a higher-level language built atop Knuth's TeX. (The names are pronounced *tek* and *lah-tek*.)

The first equation in the first paragraph on this page is encoded in LaTeX as follows:

```
\nabla \times \mathbf{E}=
  -\frac{\partial \mathbf{B}}
        {\partial{t}}.
```

Each of the terms beginning with a backslash is a LaTeX command. For example, `\frac{}{}` constructs a fraction, with whatever appears inside the two pairs of brackets forming the numerator and the denominator. Some commands simply insert a single character, as with `\times` ($\times$), `\partial` (∂) and `\nabla` (∇). The `\mathbf{}` command applies a boldface font to the bracketed text.

Let me call attention to what is *not* present in the TeX encoding. There are no explicit instructions for arranging the symbols on the page; all of the geometric details, such as centering the numerator over the denominator and drawing a horizontal bar between them, are handled automatically.

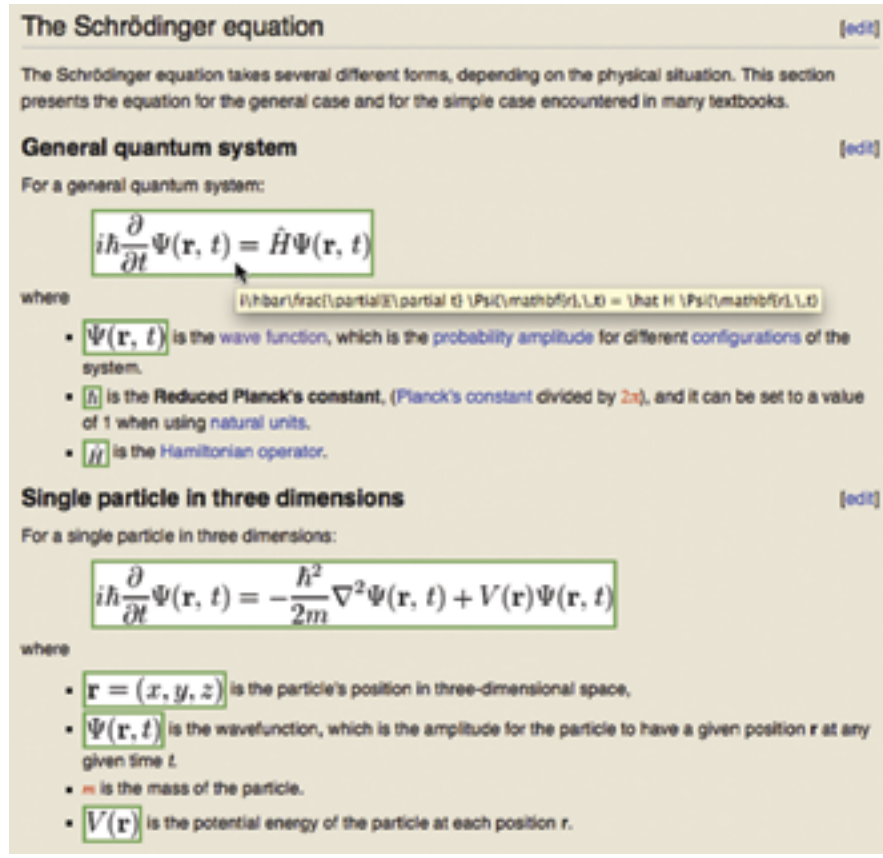
Also missing is any hint of what the equation means; TeX is a language for describing mathematical *notation*, not for doing mathematics. For example,

Brian Hayes is senior writer for American Scientist. *Additional material related to the "Computing Science" column appears in Hayes's blog at bit-player.org. Address: 211 Dacian Avenue, Durham, NC 27701. Internet: brian@bit-player.*

"$\nabla \times \mathbf{E}$" is merely a sequence of three symbols, with no reference to the operation those symbols denote—namely the "curl" of a vector field, a measure of the field's rotation or angular momentum. (The first equation is one of James Clerk Maxwell's four famous equations describing the electromagnetic field. The second example gives two definitions of the Riemann zeta function, revealing a remarkable identity between an infinite sum and an infinite product.)

TeX has transformed the process of putting mathematical ideas on paper. What once required the services of a skilled typesetter can now be done by a mere mathematician. A manuscript can go from the author to a printed journal without human intervention.

But what about the Web, which did not yet exist when Knuth developed TeX? Modern TeX systems produce output in the form of Postscript or PDF files. Although documents in these formats can be distributed over the Web—the arXiv preprint server dispatches thousands of them every day—they are not quite first-class citizens of the Web world. Browsers cannot display Postscript or PDF files without the aid of extra software, and many readers choose to download such files and view them offline or print them. This works well enough for static documents such as journal articles, but it's not ideal for the more volatile and interactive areas of the Web, such as blogs or the always-under-revision pages of Wikipedia. For mathematics to be fully at home online, it needs to be translated into one of the Web's native languages.



Mathematics remains a marginal participant on the World Wide Web, where finely typeset equations are difficult to produce and their appearance blends poorly with other textual elements. In this excerpt from a page of Wikipedia, the collaborative encyclopedia, most mathematical notation is rendered by means of embedded images, which differ in size, style and alignment from the rest of the text. In order to show how the page was assembled, the stylesheet governing the display was altered to give all images a green border and a contrasting background. A few bits of mathematical notation, marked in bright red, are not images but ordinary characters; yet even they do not match the size of the main text. The highlighted panel associated with the first equation shows the TeX code that produced the equation.

Marking It Up

From the outset, the primary language of the Web has been HTML (Hypertext Markup Language), in which "tags" identify various parts of a document's structure, such as headings, paragraphs and lists. In its earliest versions, HTML was quite simple; it couldn't even do subscripts and superscripts, so there wasn't much hope of displaying elaborate mathematical notation.

Several revisions later, HTML does have subscript and superscript tags. And a supplementary language called CSS (Cascading Style Sheets) allows finer control over many aspects of the appearance of a Web page. Modern browsers are also equipped with an interpreter for JavaScript, a programming language, so that Web pages become not just static documents but interactive programs. Still, none of these features directly address the needs of scientific and mathematical writing. There are no HTML tags for integrals, say, or for matrices.

Two main impediments stand in the way of presenting mathematics on the Web. First is the alphabet problem: Mathematicians have created a sprawling zoo of novel symbols and embellished or transformed versions of familiar characters. The nabla (∇) that appears in Maxwell's equations is a notable example, and it is joined by hundreds of other unusual glyphs—∂, $\mathbb{R}$, $\forall$, $\exists$, $\perp$, $\int$—not to mention the entire Greek alphabet and occasional borrowings from Hebrew and other languages. The difficulty of reproducing these characters has been alleviated to some extent by the recent adoption of Unicode fonts, which have room for a larger collection of glyphs than earlier font formats. But it's still not to be taken for granted that every reader will have the necessary fonts installed.

The second problem is one of layout. Mathematical notation is two-dimensional; in order to represent a matrix, say, or a summation with upper and lower bounds, it's necessary to specify the exact x and y coordinates of symbols. Some elements of mathematical notation, such as brackets and the radical that designates a square root, vary in shape and size as well as position. Encoding such geometric information in HTML and CSS is not impossible, but it stretches the technology to its limit.

Early in the history of the Web, a group of mathematicians and other interested parties gathered to address this issue in a systematic way. The result was a new markup language called

MathML, which was endorsed in 1998 by the World Wide Web Consortium. I'll return below to the present status and future prospects of MathML, but it will suffice for now to note that most of the mathematical notation to be found on the Web is *not* encoded in MathML. Instead it relies on a variety of ingenious but ad hoc workarounds. Most often there is a TeX system somewhere in the background.

One common strategy is to convert a mathematical expression to an image, or "bitmap." In some cases each symbol becomes a separate image, and multiple images have to be assembled and carefully positioned to represent an equation. In other cases an entire equation is encapsulated in a single image. The practice takes us back to the pre-alphabetic tradition of pictographic writing.

Pictograms have one big advantage: Any symbol, no matter how arcane, can be displayed in any browser. But there are also drawbacks. Symbols in the images may not blend well with typefaces on the page, and it's hard to control size, spacing and alignment. A math-intensive document could have hundreds of small images, which are slow to load and display. Images cannot be copied and pasted in the same way that text can, and the equations cannot easily be edited or revised.

But relying on fonts to supply mathematical symbols also has hazards. The author of a Web page cannot know what fonts are available on the reader's computer, or which of the available fonts will be selected for any given symbol. Thus a page that looks fine to the author may display very differently for some readers, or could be completely indecipherable.

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

TeX encoding

```
x = \frac{-b \pm \sqrt{b^2 - 4ac}} {2a}
```

MathML presentation encoding

```
<math>
  <mi>x</mi>
  <mo>=</mo>
  <mfrac>
    <mrow>
      <mo>-</mo>
      <mi>b</mi>
      <mo> </mo>
      <msqrt>
        <mrow>
          <msup>
            <mi>b</mi>
            <mn>2</mn>
          </msup>
          <mo>-</mo>
          <mn>4</mn>
          <mi>a</mi>
          <mi>c</mi>
        </mrow>
      </msqrt>
    </mrow>
    <mrow>
      <mn>2</mn>
      <mi>a</mi>
    </mrow>
  </mfrac>
</math>
```

MathML content encoding

```
<math>
  <apply>
  <eq/>
    <ci>x</ci>
    <apply>
    <divide/>
      <apply>
      <plus/>
        <apply>
        <minus/>
          <ci>b</ci>
        </apply>
        <apply>
        <root/>
          <apply>
          <minus/>
            <apply>
            <power/>
              <ci>b</ci>
              <cn>2</cn>
            </apply>
            <apply>
            <times/>
              <apply>
              <times/>
                <cn>4</cn>
                <ci>a</ci>
              </apply>
              <ci>c</ci>
            </apply>
          </apply>
        </apply>
      </apply>
      <apply>
      <times/>
        <cn>2</cn>
        <ci>a</ci>
      </apply>
    </apply>
  </apply>
</math>
```

Markup languages provide a one-dimensional description of two-dimensional mathematical notation; the markup also encodes troublesome symbols (such as the square-root sign, or radical) in an ordinary alphabet. Here the quadratic formula *(top)* is represented first in the TeX markup language and then in two versions of MathML, a language devised explicitly for displaying mathematics on the Web. The presentation layer of MathML focuses on the notation itself—on symbols and their arrangement—whereas the content layer attempts to capture meaning. Although the content description is quite prolix, it cannot quite represent the full semantics of the quadratic formula, because content MathML has no "±" operator.

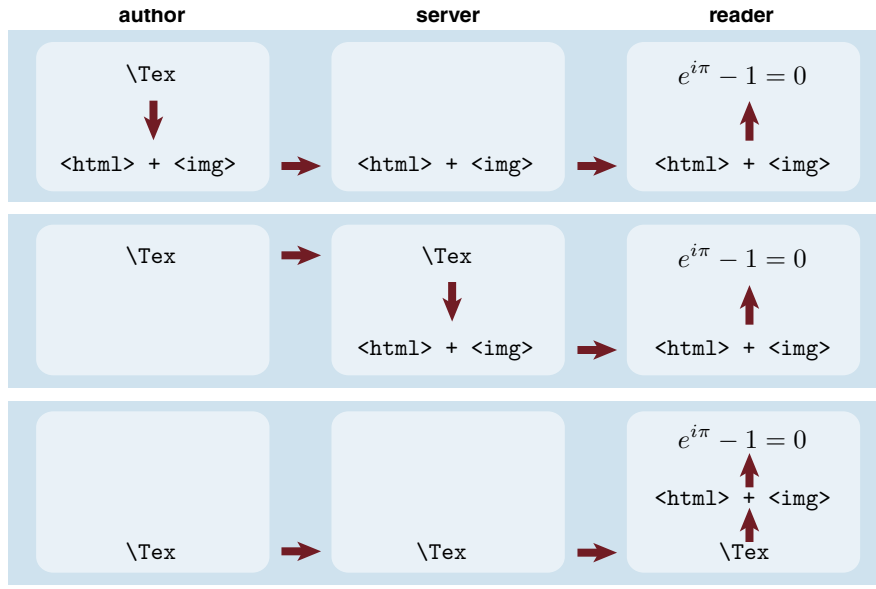
From Author to Server to Reader

For mathematical prose written in TeX, one Web-publishing strategy is to translate the entire document in advance, creating an HTML version that can then be displayed in a browser without further special handling. Two programs for this purpose are LaTeX2HTML (written in 1995 by Nikos Drakos, now of Gartner Research) and TeX4HT (by Eitan M. Gurari of Ohio State University). In essence, the translation programs replace the "back end" of a TeX system with software that generates HTML and bitmap images rather than output intended for printing.

A translator of this kind runs on the author's own computer; the HTML files and accompanying images are then uploaded to a Web server for distribution. This work flow is suited to long-lived and seldom-modified documents written in TeX. The arrangement is less convenient with frequently updated content or Web sites maintained by multiple authors in collaboration. It's also not ideal for documents that are written mostly in HTML rather than TeX, with just occasional snippets of mathematics.

A second strategy is to perform the translation not on the author's computer but on the Web server. Two programs created by John Forkosh adopt this *modus operandi*. MathTeX relies on a separate TeX system to do the actual rendering of mathematical notation; MimeTeX is self-contained, with its own TeX interpreter. Both programs generate bitmap images of entire equations. This scheme is not meant for processing complete TeX documents; it works with HTML documents that have interspersed TeX expressions. Processing mathematics on the server works particularly well for collaborative Web sites, since the translation software has to be installed on only one machine. Wikipedia and many similar sites rely on this approach. (The math processor for Wikipedia is a program called texvc, which generates images for complex expressions but outputs HTML for simple ones.)

The drawbacks of server-side software are those that afflict all image-based solutions—clumsy typography and a profusion of tiny image files. But if the result falls short of elegance, at least it works reliably for most readers, no matter what browser they choose.



Translation from one markup language into another and finally into a formatted equation takes place in stages that can be performed at various points along the pathway from author to reader. Typically, an input language such as TeX is converted into some combination of HTML (the main language of the Web) and images. The bulk of this work can happen on the author's computer *(top)*, on the Web server *(middle)* or on the reader's computer *(bottom)*.

Let the Browser Do It

As we have just seen, translation software can run on the author's computer or on a server. There is one more possibility: performing the translation on the reader's computer, or what is known as "the client side." Under this plan, TeX or some other encoding of mathematical content is written into the Web document and passed along unchanged by the server, to be interpreted by the browser.

Of course the problem with sending TeX to the browser is that the browser has no idea what to do with it. That issue can be addressed by means of a plug-in—a software component installed within the browser.

The best known mathematical plug-in is techexplorer, a program initially developed by Robert S. Sutor at IBM and now maintained and distributed by Integre Technical Publishing. Techexplorer can display complete LaTeX documents or it can render just the mathematical expressions within an HTML document. In either case the display of equations is based on fonts rather than images.

The big advantage of a software plug-in is that the rendering of mathematics is liberated from all the constraints of HTML. The plug-in can supply its own fonts and can place characters with as much precision as the hardware will allow. The big *dis*advantage is that none of this magic works until the end user downloads and installs the plug-in. This barrier to entry tends to discourage casual visitors to a Web site. It also creates a threshold effect: Authors hesitate to rely on the technology until enough readers adopt it, and vice versa.

An interesting alternative to a plug-in might be called a slip-in. The idea is to bundle up the translator program and include it as part of the Web page itself. There's no need for the user to install any software; the translation program runs automatically when the page is loaded into a browser. A program of this kind called jsMath takes advantage of the JavaScript programming language built into Web browsers.

JsMath is the creation of Davide P. Cervone of Union College in Schenectady, N.Y. Cervone undertook the project mainly to meet his own needs: He wanted to distribute class notes and homework assignments via the Web, and none of the available solutions were entirely satisfactory. So he wrote what amounts to a TeX interpreter in a JavaScript program.

JsMath offers three styles of equation rendering. The most graceful display requires a set of six fonts based on the Computer Modern faces introduced by Knuth; those fonts are freely available, but jsMath can access them only if the reader downloads and installs them. A fallback strategy is to assemble equations from individual character images. The full set of images—some 78 megabytes worth—is stored on the server; only the subset needed is downloaded with any particular document. The third option is to build equations from Unicode fonts. The reader selects one of the three rendering methods through a pop-up control panel.

Squeezing a TeX interpreter into a Web page is an impressive feat, but it adds considerable bulk and complexity. Documents with many equations take a while to finish rendering. Cervone is now launching a follow-on project called MathJAX, supported by several publishers of mathematical software. The aim is to make the system more flexible and responsive.

Whatever Happened to MathML?

Typesetting mathematics with HTML and bitmap images is rather like turning a bicycle into a sailboat. You can't help admiring the audacity of the attempt, but the result still doesn't seem like the best vehicle for the purpose. Why not choose MathML, which was designed explicitly for this task?

MathML is a variety of XML (the eXtensible Markup Language). Compared with TeX, it is a more formal

language, and it is also far more verbose. Consider the simple expression $x+1$, which might be encoded in TeX as `$x+1$`. (The dollar signs mark the content as mathematics rather than ordinary text.) In MathML the same expression takes this form:

```
<mrow>
  <mi>x</mi>
  <mo>+</mo>
  <mn>1</mn>
</mrow>.
```

Each symbol is tagged to indicate its role—<mi> for an identifier, <mo> for an operator, <mn> for a number—and the expression as a whole is wrapped in an <mrow> tag to show that it belongs on a single line. Capturing this information is potentially useful, since identifiers, operators and numbers are accorded different typographic treatment. (TeX has to infer the role of each symbol, and occasionally gets it wrong.)

There's more. The style of markup shown above is only half of MathML. It's called the presentation language; there is also a content language, which attempts to express meaning rather than layout. The expression $x+1$ would have this content markup:

```
<apply>
  <plus/>
  <ci>x</ci>
  <cn>1</cn>
</apply>.
```

Here <plus/> does not refer to the symbol + but to the mathematical operation of addition. In this structure we get a glimpse of a grand vision—Web pages with *active* mathematical content, where the menu of things you might do with an expression includes not just copying, pasting and printing but also solving an equation, graphing a function and factoring a polynomial.

MathML has an enthusiastic community of developers and users. There is commercial software for writing and editing MathML documents (notably from Design Science and Integre Technical Publishing) as well as a noncommercial translation program called ASCIIMathML, created by Peter Jipsen of Chapman University in Orange, Calif. Several large scholarly publishers, including the American Institute of Physics, have based their operations on XML and MathML; so has the U.S. Patent Office.

On the Web, however, MathML has not exactly swept away the competition. One reason is lack of support in browsers. In the early years, the only way to read MathML Web documents was with plug-in software. More recently a few browsers—notably those of the Mozilla family, such as Firefox—have gained native support for MathML. But there is still confusion over how MathML content should be embedded in an HTML document. Moreover, MathML has not solved the fonts problem; readers are still responsible for installing appropriate fonts.

Another factor inhibiting the spread of MathML is simply that TeX is deeply entrenched, particularly in physics, mathematics and computer science. If you live in a TeX-centric universe—I have a friend who even writes love letters in TeX—it's hard to see any benefit of a new and very different language.

Embedded Assets

How will it all turn out? Will Web sites of the future be chock full of MathML, or will TeX and HTML continue to prevail? Or will something else altogether come along?

I have no answers for these questions, but I want to suggest an adjustment in the way the Web works—a small change that could improve *any* strategy for displaying mathematical notation. It has to do with where fonts come from.

Under the present rules, a Web author can request a particular font, and the reader's browser will honor the request if the font is available on the client machine. If not, some default font is substituted. Wouldn't it be more helpful if the author could supply the missing font, either by embedding it directly in the page or by referring the browser to a site where the font is available? Given such a mechanism, any font-based system for presenting mathematics could ensure that all the needed symbols are ready at hand.

This is not a new idea. A proposal for "Webfonts" was included in a draft CSS standard in 1998, and the idea was even implemented in a few browsers, including Microsoft's Internet Explorer. But the proposal never caught on, and it was removed from later versions of the standard. Recently, Håkon Wium Lie of Opera Software has called for renewed consideration of the idea.

Much of the discussion centers on legal questions of interest to the owners of typeface copyrights. This doesn't seem like an insuperable problem. It was solved in the case of PDF files, which do embed fonts. Even if proprietary typefaces were off limits, there are enough freely available fonts—including all those commonly used with TeX—to make the prospect attractive.

For a change of this kind to have any impact, all the browser makers would have to adopt it. Those are the same people who have so far resisted implementing MathML. Why would they treat the font proposal any differently? I think there is reason for optimism on this score, not because the mathematical community has much clout but because embedded fonts would be of value to other constituencies. Advertisers, in particular, would be pleased to gain greater control over Web typography.

Meanwhile, as I finish preparing this column for the press, I also face the task of helping to get my own penalty copy ready for publication on the *American Scientist* Web site. Those irksome equations and curious characters I've been writing about will somehow have to be made Web-friendly. I don't know exactly how we're going to do that, but I suspect some bicycles are going to be outfitted with spinnakers and jibs.

Bibliography

Beeton, Barbara, Asmus Freytag and Murray Sargent III. 2008. Unicode support for mathematics. Unicode Technical Report No. 25. http://www.unicode.org/reports/tr25

Bos, Bert. 2008. For and against standardizing font embedding. http://www.w3.org/Fonts/Misc/eot-report-2008

Cervone, Davide P. Web site. jsMath: A method of including mathematics in Web pages. http://www.math.union.edu/~dpvc/jsMath

Goossens, Michel, and Sebastian Rahtz with Eitan Gurari, Ross Moore and Robert Sutor. 1999. *The LaTeX Web Companion: Integrating TeX, HTML, and XML*. Reading, Mass.: Addison-Wesley Longman.

Jipsen, Peter. Web site. Translating ASCII math notation to Presentation MathML. http://www1.chapman.edu/~jipsen/asciimath.html

Knuth, Donald E. 1984. *The TEXbook*. Illustrations by Duane Bibby. Reading, Mass.: Addison-Wesley Pub. Co.

Miner, Robert, and Jeff Schaefer. 1998. A gentle introduction to MathML. http://www.dessci.com/en/support/mathtype/tutorials/mathml/default.htm

Miner, Robert. 2005. The importance of MathML to mathematics communication. *Notices of the AMS* 52:532–538.

Soiffer, Neil. 1997. MathML: A proposal for representing mathematics in HTML. *SIGSAM Bulletin* 31(3):44.

Wright, Francis J. 2000. Interactive mathematics via the Web using MathML. *SIGSAM Bulletin* 34(2):49–57.

AmSci May/June 2009 Vol 97 No 3 p. 181

rows Bridge. A report by investigating engineers concluded that wind was the enemy of suspension bridges—something Roebling learned a century earlier. Later engineers evidently did not recognize that it was relevant to their modern structures.

The Bronx-Whitestone Bridge was among several built in the late 1930s, when aesthetic goals drove design, that had wind troubles. A stiffening truss was added to that bridge—obstructing a great view of the Manhattan skyline—but that truss is no longer in place. A few years ago, fairings designed to ameliorate the wind effects on the plate girders, along with other motion-checking and –damping devices, were installed to steady the deck.

Illustration Credits

Computing Science
Pages 187–190 Brian Hayes

The Origin of Life
Figures 2–4, pages 209–211
Barbara Aulicino and Morgan Ryan
Figure 5 *(bottom right)*
Barbara Aulicino

Fermi, Pasta, Ulam and the Birth of Experimental Mathematics
Figures 2–4, 8 *(left)* Barbara Aulicino

The Deprived Human Brain
Figures 2, 3, 5 Stephanie Freese

Revisiting the Limits to Growth After Peak Oil
Figures 5, 8, 10 Stephanie Freese
Figures 7, 9 Barbara Aulicino

Wrong Keyes?

To the Editors:

I haven't read *American Scientist* for a while and found much of interest in the March–April issue. On page 90, David Schoonmaker cites "William W. Keyes." This sounded very much like my colleague Robert W. Keyes, and indeed he is so identified on page 134.

Richard L. Garwin
Scarsdale, NY

How to Write to *American Scientist*

Brief letters commenting on articles that have appeared in the magazine are welcomed. The editors reserve the right to edit submissions. Please include a fax number or e-mail address if possible. Address: Letters to the Editors, American Scientist, P.O. Box 13975, Research Triangle Park, NC 27709 or editors@amscionline.org.

Erratum

In "Writing Math on the Web" by Brian Hayes (March–April), the equation in the illustration on page 101 should have been $e^{in} + 1 = 0$.