

Using TeXShop 5.0 to Write Interactive Documents

Richard Koch

August 26, 2022

1 Introduction

TeXShop 5.0 contains several new features. I'm going to illustrate how these features can be used to write course notes and similar documents with interactive content.

I'll use an old project I happen to have around the house. The name of the document is *Fourier*. Please open this document and follow along as I explain the new TeXShop features.

The entire session will be done within TeXShop. I'll use only three other applications for very minor reasons, and I'll point out those moments.

2 A Document about Fourier Series

Fourier is a mathematics document. If you are into math, it introduces Fourier series and contains Dirichlet's 1829 proof that the Fourier series of a piecewise differentiable function converges pointwise to the function. The initial line of this document is a comment, but TeXShop understands that comment. It instructs TeXShop to typeset the document using pdflatex.

Start by opening the document and typesetting. Notice that TeXShop works as it always has. Scroll through the output. The document contains illustrations and extensive mathematics; it uses hyperref and other standard packages, and has a table of contents. The line activating the table of contents is commented out to make the document easier to read. Feel free to activate that line. The standard TeXShop view of such a document is shown at the top of the next page.

Ready for experiments?

Fourier.tex

```

1 % ITEX TS-program = TeX4ht
2
3 \pdfminorversion=4
4 \documentclass[11pt, oneside]{amsart}
5 \usepackage{geometry}           % See geometry.pdf
6 \geometry{letterpaper}         % ... or a4paper
7 %\geometry{landscape}          % Activate for
8 \usepackage[parfill]{parskip}   % Activate to begin
9 \usepackage{graphicx}
10 \usepackage{amssymb}
11 \usepackage{amsmath}
12 \usepackage{epstopdf}
13 \usepackage{float}
14 \usepackage{iftex}
15 \DeclareGraphicsRule{.tif}{png}{.png}{convert #1}
16
17 \usepackage[colorlinks=true, pdfstartview=FitV, linkcolor=blue, citecolor=blue]{hyperref}
18
19
20 \newtheorem{theorem}{Theorem}
21 \newtheorem{corollary}[theorem]{Corollary}
22 \newtheorem{definition}[theorem]{Definition}
23 \newtheorem{lemma}[theorem]{Lemma}
24 \newtheorem{exercise}[theorem]{Exercise}
25 \newtheorem{remark}[theorem]{Remark}
26 \newtheorem{example}[theorem]{Example}
27 \newtheorem{warning}[theorem]{Warning}
28 \def\Re{\mbox{Re}}
29
30 \title{Dirichlet's Theorem}
31 \author{Richard Koch}
32 %\date{}                      % Activate to
33 \begin{document}
34 \maketitle

```

Fourier.pdf

DIRICHLET'S THEOREM
RICHARD KOCH

1. WAVES

A wave is a periodic function $f(x)$. This means that there is a fixed p , called a *period* of f , and for all x we have $f(x+p) = f(x)$. We can change the period by an easy renormalization, so from now on all waves have period 2π .

Obviously $\sin(x)$ is periodic; that is why we fixed 2π as the period. Notice that $\sin(kx)$ also has this period for any integer k .

Given one wave f , we can obtain new waves by translation. This is called *changing the phase*. Thus $f(x - \delta)$ has phase δ .

The sum of waves is another wave. So we can obtain very general waves using formulas of

3 Typesetting with TeX4ht

In the first line, change the word “pdflatex” to “TeX4ht”. This tells TeXShop to typeset with the engine TeX4ht, a program by Eitan Gurari that accepts standard LaTeX as source, but outputs a web page in html. Notice that the change has been made in the image above. Close the Preview window and then typeset. The result is shown at the top of the next page.

The TeX4ht engine typesets twice, once with pdflatex and once with TeX4ht. The pdf output is opened in a standard Preview window and the html output is opened in a TeXShop web view. Thus three windows are visible.

The image shows a LaTeX editor, a PDF viewer, and a web browser all displaying the same document. The LaTeX editor shows the source code, the PDF viewer shows the rendered mathematical plots, and the web browser shows the HTML version of the document.

Scroll through the html document. It is essentially perfect. The illustrations look fine and the mathematical equations are crystal clear. It is difficult to believe that this is an html document. To make sure, resize the pdf window and notice that the contents shrink (if you are in “fit to window mode”). Resize the html and notice that the text reflows.

To be completely honest, there are a couple of minor flaws in the html. Some of the graphic images are too narrow. There is an easy fix for this problem, but I deliberately did nothing because I want to show that TeX4ht is almost perfect. One piece of mathematics failed to appear. I explained that a Fourier series converts a wave into a fundamental note and various harmonics. If the note is a low C , the harmonics will start an octave higher and contain the notes $C, G, C', E', G', B^b, C'', D'', \dots$ TeX4ht didn’t like that flat sign.

Fifteen years ago I watched Eitan demonstrate TeX4ht at a TUG Conference. At the time, TeX4ht converted mathematical equations into small pictures and displayed those pictures. The output was a little crude. I thought TeX4ht was an amusing program, but completely misunderstood its significance or the difficult problems that Eitan had solved.

In 2009, just weeks before he was giving another TUG talk, Eitan unexpectedly died. But TUG paid him the ultimate compliment by keeping his program alive. The current programming is done by Michal Hoftich, who is very active with updates almost weekly.

Eitan's hard work took wings due to two independent and enormous projects: MathML and MathJax.

The web was invented on a NeXt computer by Tim Berners-Lee, a physicist working at CERN. It could do many wonderful things, but surprisingly it could not display mathematical content. This was eventually solved by the invention of MathML, a new tagging scheme able to encode mathematical equations. It was not sufficient to invent the scheme; browsers had to be revised to render the equations. This took many years. I heard several early talks about MathML at TUG and my reaction was "luckily, I'll never have to use that."

Safari and most other browsers can render MathML, but it is not perfect. Integral signs are often far smaller than they should be and there are other flaws. This problem was definitively solved by another project called MathJax, which searches an html document for its MathML content and directly renders that content, bypassing the browser.

MathJax began as a project by Davide Cervone in 2004, when it was called jsMath. Eventually the American Mathematical Society recognized the importance of this project, and designed the MathJax project in 2009 as a successor to jsMath.

Both MathML and MathJax are enormous projects. Please don't ask me how they work because I have no idea. Luckily, the projects are easy to use, and TeXShop's support mainly happened automatically.

4 Why is HTML Important?

Some students have desktop computers, some have portables, some had iPads or other devices. All devices have powerful browsers and web support, and all can display pdf documents. But pdf is a static format, so if a screen is small, the only choice is make all the text smaller. Html can reflow the text to work well on multiple screen sizes.

Html is interactive. Students can enter text into a document, or check boxes, or select answers from a list, or run experiments. Pdf is a static format. Attempts have been made to add interaction to pdf, but progress is slow and uneven across platforms.

Much recent work has been devoted to making documents accessible to readers with disabilities. The area is complicated because a system useful with one disability may be irrelevant for those with a different one. Some members of TUG have made a point of contacting actual users to understand their needs and preferences. One member reported that his contact preferred the tagging system on html documents to those on pdf documents, and avoided pdf documents if possible.

Who knows what is best? We don't have to choose. We can output documents in both pdf and html forms. Put both on the web, and let your users decide which to use.

5 A Miscellaneous Feature

Activate the Pdf Preview Window. Scroll to section 9 in the document. The text reads “Here is an entirely irrelevant link to the slides of a talk I once gave at Cal Poly Humboldt in Arcata, California.” And indeed this is a url to a series of slides on my University of Oregon web site. The slides have nothing to do with Fourier Series. Click the link anyway.

Earlier I warned that I would run three external programs. This is one of those times. Safari opens and displays the link. That’s how links work in a pdf document. Quit Safari.

Now find section 9 in the html document. The ninth section has a different name in this document. We’ll come back to that later. But the previous link is there. Click it. This time the link opens directly in the html preview window, because that’s how the web works. We aren’t distracted by another program. Use the arrows in the toolbar to return to the original document.

The left window shows a PDF document titled "Fourier.pdf". It contains a mathematical derivation of a Fourier series approximation. The formula shown is:

$$b_k = \frac{2}{\pi} \int_0^{\pi} \sin kx \, dx = \frac{2}{k\pi} (-\cos kx) \Big|_0^{\pi} = \frac{2}{k\pi} (1 - \cos k\pi)$$

So all terms are zero except $b_k = \frac{4}{k\pi}$ for k odd

$$f(x) = \frac{4}{\pi} \left(\sin x + \frac{1}{3} \sin 3x + \frac{1}{5} \sin 5x + \dots \right)$$

Substitute $x = \frac{\pi}{2}$ in this formula to obtain famous result

$$\frac{\pi}{4} = 1 - \frac{1}{3} + \frac{1}{5} - \dots$$

9. FIRST

Here is an entirely irrelevant link to the slides in Arcata, California: <https://pages.uoregon.edu/farlow/ma261/ma261.html>

10. DIRICHLET

Dirichlet is one of my heroes. His wife was a siste in his home. After Gauss convinced Riemann and study mathematics, Gauss sent him to E Dirichlet had an enormous influence in all of I

In number theory, Dirichlet proved that infinite written in base 10, and proved the correspondi been used by Legendre in an attempted proof c complained that the result was considerably de it. But Dirichlet proved that considerably dec proof of this result is desirable, particularly si

The right window shows a slide from the WWDC 2011 featuring OSX Lion, iOS 5, and iCloud. The slide has a large banner with the text "WWDC, Apple's Worldwide Developer's Conference" and images of a lion and the iOS 5 logo. Below the banner, several people are standing in a large hall, likely the conference center.

6 Different Output

Notice that section 9 looks completely different in the pdf and html documents. Even the name of the section is different. Here are the two versions of section 9:

Substitute $x = \frac{\pi}{2}$ in this formula to obtain $1 = \frac{4}{\pi} \left(1 - \frac{1}{3} + \frac{1}{5} - \dots\right)$ and thus Leibniz' famous result

$$\frac{\pi}{4} = 1 - \frac{1}{3} + \frac{1}{5} - \dots$$

9. FIRST STEPS

Here is an entirely irrelevant link to the slides of a talk I once gave at Cal Poly Humboldt in Arcata, California: <https://pages.uoregon.edu/koch/HumboldtTalk.pdf>

10. DIRICHLET

Dirichlet is one of my heroes. His wife was a sister of Mendelsohn, so there was a lot of music

Substitute $x = \frac{\pi}{2}$ in this formula to obtain $1 = \frac{4}{\pi} \left(1 - \frac{1}{3} + \frac{1}{5} - \dots\right)$ and thus Leibniz' famous result

$$\frac{\pi}{4} = 1 - \frac{1}{3} + \frac{1}{5} - \dots$$

9. Interactive Experiments

Here are interesting experiments. [Interactive/Page1.html](#)

This is the Minimal example from the PreTeXt distribution. [Interactive/Minimal/minimal.html](#)

Here is an entirely irrelevant link to the slides of a talk I once gave at Cal Poly Humboldt in Arcata, California: <https://pages.uoregon.edu/koch/HumboldtTalk.pdf>

10. Dirichlet

Dirichlet is one of my heroes. His wife was a sister of Mendelsohn, so there was a lot of music in his home. After Gauss convinced Riemann to drop his plans to get a theology degree and study mathematics, Gauss sent him to Berlin for two years to work with Dirichlet. Dirichlet had an enormous influence in all of Riemann's work.

What is the LaTeX source code which produced this result? It is easy to find that source because synctex works in the pdf Preview, and we discover the following in the LaTeX source code:

```

\ifx\HCode\undefined
    \section{First Steps}
\else
    \section{Interactive Experiments}
    Here are interesting experiments. \url{Interactive/Page1.html}

    This is the Minimal example from the PreTeXt distribution.
    \url{Interactive/Minimal/minimal.html}
\fi

Here is an entirely irrelevant link to the slides of a talk ...

```

The first line in this source is an obscure fact from the TeX4ht world; that line is false if TeX4ht is running, and true in all other cases. Consequently, it is easy to produce different output in the pdf document and the html document.

7 The Point of It All

We now come to the key moment of this demonstration. Click the link to Page 1 in the html document. An entirely new view opens in the html Preview window. That page is shown at the top of the next page. I was too lazy to write a page related to Fourier series, so this page contains miscellaneous things. Forgive me.

Page 1 is a confusing page. It has elements which suggest that it was written directly in html, but the mathematics suggests that it came from a LaTeX source.

Luckily, there is an easy way for us to find the source. Go to the pdf Preview page, and click on the page title at the top while holding down the control key. A menu drops down showing the various folders in the path to the source file for Fourier. Click the top item in the list and the Finder will open the folder containing the source. In that source you will find a folder named Interactive, and in that folder you will find a file named Page1.html. Drag this file to the TeXShop icon and drop it there to open Page1.html in a TeXShop edit window. (Incidentally, this is the second of three uses of external applications. We used the Finder briefly.)

Once the source is open, there is absolutely no doubt that our page is written in html.

Fourier.html

URL: Search

Writing Mathematical Formulas; Elementary Constructions

When $a \neq 0$, there are two solutions to $ax^2 + bx + c = 0$ and they are

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

Also

$$\int_0^\infty e^{-x^2} dx = \frac{\sqrt{\pi}}{2}$$

And $\alpha, \beta, \Gamma_{j_1 j_2}^{i_1}$.

This is an example
And so is this
And also this

We can have **bold type** and *italic text* and stuff like that.

Calling Sage

Sage is a fantastic project from the University of Washington to provide an open source program able to do mathematical calculations and display the result. A server exists to perform Sage calculations over the internet. Thus none of the examples below require a local copy of Sage. These examples were copied from Sage documentation; see <https://wiki.sagemath.org/>

and also <https://www.sagemath.org>

Your own computations

Type your own Sage computation below and click “Evaluate”.

```
1 plot(sin(x), (x, 0, 2*pi))
```

But TeXShop 5 can process html files, so we are still in business. Indeed looking at the top line of Page1.html, we see that TeXShop can apparently typeset this source file. Go ahead and typeset. Notice that a new html window opens showing exactly the same page as in the html version of Fourier. But there is a difference in these pages. The new html window only contains the output from this html source, and the back and forth arrows in the toolbar do nothing. But the Fourier html window’s arrows move between the Fourier document and this new window.

It turns out that when we edit the html source file, the change appears in both html preview

windows. For the moment, close the source we created by typesetting, and use the Fourier document's html window.

8 The Interactive Page

At this point, you should concentrate on two windows, the window containing the source for Page1.html and the Fourier html Preview showing the resulting Page 1. See the image below. We are going to study various items on these pages.

The image shows the Fourier software interface with two windows. The left window is titled 'Page1.html' and is in the 'Typeset' tab, showing the raw HTML code for a mathematical document. The right window is the 'Page1.html' preview, showing the rendered content. The preview window contains the following text:

Writing Mathematical Formulas; Elementary Constructions

When $a \neq 0$, there are two solutions to $ax^2 + bx + c = 0$ and they are

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

Also

$$\int_0^\infty e^{-x^2} dx = \frac{\sqrt{\pi}}{2}$$

And $\alpha, \beta, \Gamma_{j_1 j_2}^{i_1}$.

This is an example
And so is this
And also this

We can have bold type and italic text

Calling Sage

Sage is a fantastic project from the University of Washington to provide an open source program able to do mathematical calculations and display the result. A server exists to perform Sage calculations over the internet. Thus none of the examples below require a local copy of Sage. These examples were copied from Sage documentation; see <https://wiki.sagemath.org/> and also <https://www.sagemath.org>

Your own computations

Type your own Sage computation below and click "Evaluate".

```
1 plot(sin(x), (x, 0, 2*pi))
```

Evaluate

The first thing we notice is that this page contains mathematical formulas which look beautiful and clear. Writing mathematics directly in html is supposed to be difficult; that

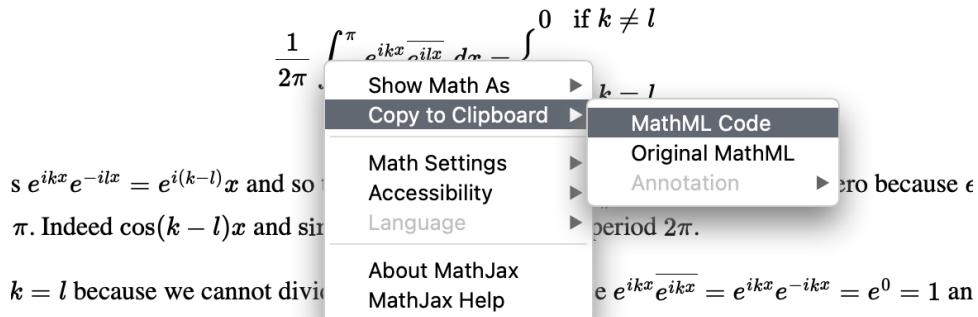
is why MathML was invented. But writing MathML by hand is impossible.

A close look at the mathematical source on the left shows that this source is written directly in LaTeX. How is that even possible?

Let us do a little experiment. Using the toolbar in the web view, flip back to the main Fourier material, and find an interesting formula. Then click on the formula while holding down the control key. A contextual menu opens offering to copy the source for the formula. As you see in the screen shot below, this menu offers two different versions of MathML code. Where did this menu come from?

5. The Orthogonality Relations

are integers,



we could just as well integrate from 0 to 2π . It is traditional to integrate from $-\pi$ to π .

6. Fourier's Amazing Formula for c_k

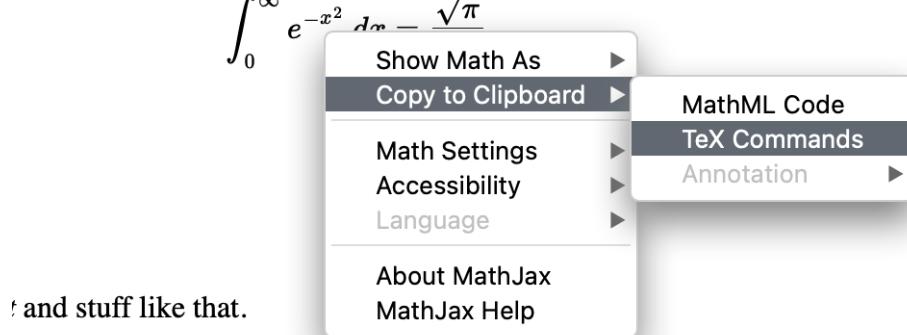
.. The c_k are given by

We can copy the result into an empty TeXShop source window, and we get a long chain of MathML commands, which start as follows:

```
<math xmlns="http://www.w3.org/1998/Math/MathML" display="block">
<mrow>
<mo fence="true" form="prefix"></mo>
<mrow>
<mfrac>
<mrow>
<mn>1</mn>
</mrow>
<mrow>
<mn>2</mn>
```

and continue in that vein for two pages.

Now let us try the same experiment for the mathematics on the interactive page. We again get the mysterious popup menu, but this time it offers to provide either MathML code or TeX code:



and stuff like that.

When we select TeX Commands, copy, and paste into an empty window, we get

```
\int_0^\infty e^{-x^2} \, dx = {\sqrt{\pi}} / 2
```

This is the LaTeX source for the formula. Moreover (and now I am going to open the final third application that I promised) I could open LaTeXiT and copy this result to the

bottom panel and LaTeX it. I get the original integral, which can be dragged to Word, or Pages, or any other application in standard LaTeXiT manner.

Where are these menus coming from, and why is there a difference?

9 More about TeX4ht

The TeX4ht program can process mathematical equations in several different ways, depending on parameters passed to it when it is called. It can provide small pictures, like the original version. It can output MathML and leave the rendering of this code to the browser. It can output MathML, but call MathJax to do the rendering. Or it can output ordinary LaTeX and call MathJax to do the rendering.

I experimented with the last three variations. Asking the browser to render MathML was the worse choice; the results were acceptable but nothing to brag about. Outputting MathML but calling MathJax to render was the option that worked best for me, so that is what the Math4ht engine calls. Outputting Latex code and asking MathJax to render also looked great, but more formulas refused to render. The TeX4ht documentation warned that there were bugs with that option, and added that some of these bugs may never be fixed because there are conflicts between the html tagging system and the LaTeX markup system.

On the other hand, when users want to directly enter html code for interactive pages, requiring them to type MathML is not an option. So for those pages, I set MathJax to render standard LaTeX code. Users are not likely to write really elaborate equations in interactive pages, and if they do, they'll immediately notice a problem and modify the code.

Therefore, in both kinds of pages the rendering is done by MathJax. And those beautiful popup menus are provided automatically by MathJax. Notice that the menus follow the Apple Design Guidelines. Presumably the MathJax programmers follow Windows guidelines when running on a windows system, and follow Linux guidelines (if such guidelines exist) on Linux machines.

10 Inspecting Typesetting Engines

There is nothing mysterious about the typesetting engines in TeXShop. All are immediately available for inspection by users. To see the TeX4ht engine, go to the TeXShop menu in TeXShop and select “Open ~/Library/TeXShop.” The Finder will show a list of folders. Open the “Engines” folder, find TeX4ht.engine, and open it in TeXShop. Inside you will find the following line:

```
make4ht "$1" "mathml,mathjax"
```

This is the line which asks TeX4ht to typeset. The final “mathml,mathjax” says to output MathML code, but then render it with MathJax. Change this to “mathml,mathml” to let the browser render the MathML.

In Engines/Inactive/Tex4ht we provide an engine named TeX4ht-MathJax.engine which causes TeX4ht to output Latex code and call MathJax to render it, if you would like to experiment with that option.

11 Html Files in the Interactive Folder

Recall that the html source for our interactive page is found at Interactive/Page1.html where Interactive is a folder in the main source directory. Nothing in TeXShop forces this design, but it is a useful way to keep the interactive pages and the regular Fourier source separate. If a document has many interactive pages, it would be best to put each source file in the Interactive folder.

Some of these interactive pages may require support files. In that case, create a sub-folder for the interactive page and put the page source and the support files in that sub-folder. So for instance, if Page2.html requires A.html and B.png, then put Page2.html, A.html, B.png in a folder, perhaps named FirstExercise, so the path to Page2 is Interactive/FirstExercise/Page2.html.

These interactive pages are completely independent of the main Fourier.html. Each is a fully independent web page. TeXShop need not create these pages. To take a particularly outrageous example, there is an independent project called PreTeXt by Robert Beezer at the University of Puget Sound. Some University of Oregon faculty use his system, and it is so well-organized that I always recommend that users examine it for inspiration. It is described in detail in the TeXShop 5.0 Change Document. PreTeXt authors write source in xml, and then the source is converted to pdf, html, and other formats. The PreTeXt system comes with several examples including a project called Minimal. I reached into this project and pulled out minimal.html, the html version of this example. I placed this file, and its support files, in Inactive/MinimalExample. In the same section 9 containing the link to Page 1, there is a second link to Minimal. Select it to see what PreTeXt’s html web pages look like.

An immediate consequence of all of this is that we can examine the source for Part1.html to see exactly how MathJax is called to interpret those LaTeX source lines. I’ll do that a little later; it isn’t difficult.

12 Sage

SageMath is an open source alternative to the computer algebra systems Magma, Maple, Mathematica, and MATLAB. The project was created by William Stein, a mathematician at the University of Washington, and released on February 24, 2005. See <https://sagemath.org> and <https://wiki.sagemath.org>. Sage is mostly written in Python, but it integrates many previous open source projects written in C, Lisp, and Fortran. Among these are Gap, GP, Macaulay, Maxima, Octave, and R. The program has been used for serious research on Elliptic Curves, Finite Groups, and many other areas, and has an active support group with contributions by thousands.

The Sage web site has an install package for the Macintosh. Sage has support for LaTeX, so it is possible to write Sage Code in the middle of a LaTeX document and automatically call Sage during typesetting to produce a graph, or compute an integral symbolically. However, the development I'll show does not depend on installing SageMath, and is independent of the facility to integrate Sage calls into a LaTeX source file.

Sage maintains a server which can run Sage over the web. This server allows web pages to contain interactive content based on SageMath. See <https://sagecell.sagemath.org> and other links from that page for details. The servers can be used for free. The web page Page1.html contains several examples, copied directly from the Sage web pages. To repeat: I did not write any of these programs myself. And to also repeat: the Page1.html web page should work for any user in the world without installing Sage.

The interactive page contains the following item:

Your own computations

Type your own Sage computation below and click “Evaluate”.

```
1 plot(sin(x), (x, 0, 2*pi))
```



Evaluate

..

When the Evaluate button is pressed, the item changes to the view at the top of the following page.

But there is more. The text “plot(sin(x), (x, 0, 2*pi))” in this item is editable, so if you change it to “plot(sin(x) + cos(3*x)/3, (x, 0, 2*pi))” and push Evaluate again, you’ll produce the second plot on the next page.

Therefore this item allows readers to experiment by plotting many graphs. In Fourier

theory, they can test the effect of changing the Fourier coefficients of expressions.

Your own computations

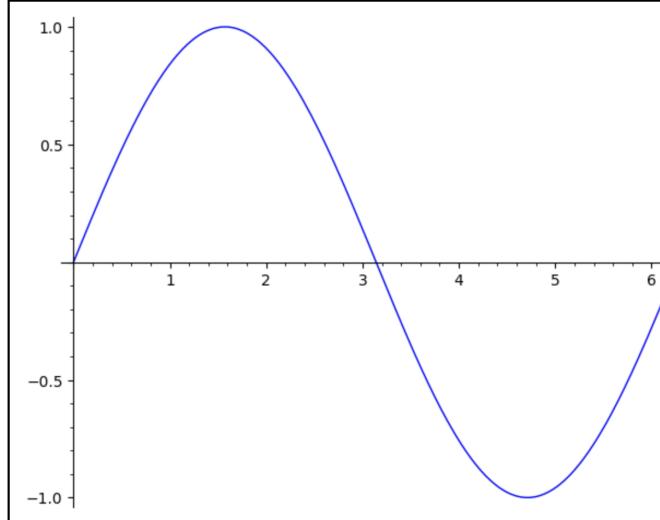
Type your own Sage computation below and click “Evaluate”.

```
1 plot(sin(x), (x, 0, 2*pi))
```



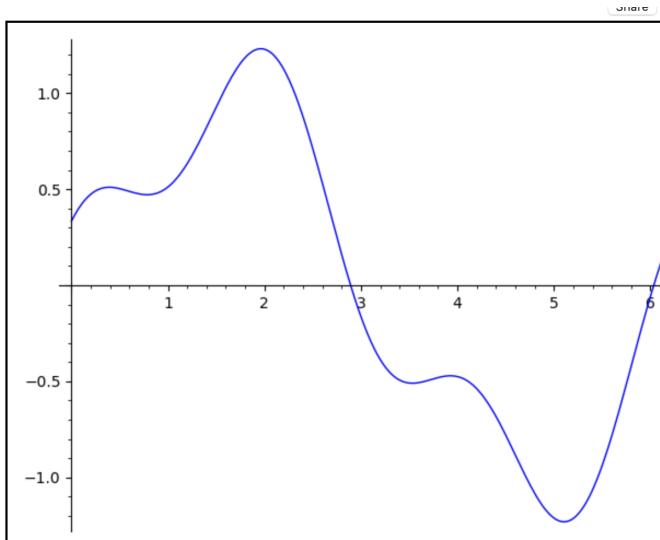
Evaluate

Share



[Help](#) | Powered by [SageMath](#)

..



[Help](#) | Powered by [SageMath](#)

..

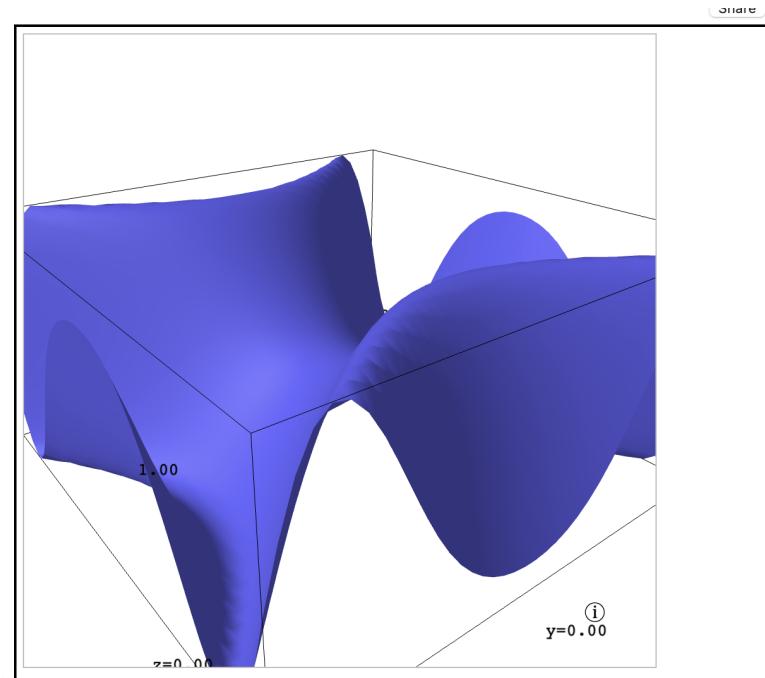
Here is the source code which generated the previous plotting example. Very simple.

```
<div class="compute"><script type="text/x-sage">
plot(sin(x), (x, 0, 2*pi))
</script></div>
```

However, the amazing thing is that the line between the first and last lines of this source can be replaced by *any Sage program*. So very complicated interactive items can be created as soon as the author of the document learns a little Sage (or, like me, just copies the work of others). Here are three examples. I confess that I find these utterly astonishing. If the Sage code is replaced with

```
x, y = var('x,y')
plot3d(sin(x^2 - y^2), (x,-2, 2), (y,-2,2))
```

we get



This picture is not all that impressive on the printed page, but if you are working with the live plot in the Fourier example, you'll notice that you can hold down the mouse button and rotate the figure in real time. This interaction could not possibly happen in real time over the internet, so Sage must be directly communicating with the GPU in the Macintosh. Since the program is platform independent, it must also have different code when the user is running Windows, or Linux with various graphic cards.

The example labeled “A Calculus Course Experience” contains the following code. I list it to show that quite complicated program can be run by the server. This code is by Nick Alexander, based on the work of Marshall Hampton.

```

var('x')
@interact
def midpoint(f = input_box(default = sin(x^2) + 2, type = SR),
    interval=range_slider(0, 10, 1, default=(0, 4), label="Interval"),
    number_of_subdivisions = slider(1, 20, 1, default=4, label="Number of boxes"),
    endpoint_rule = selector(['Midpoint', 'Left', 'Right', 'Upper', 'Lower'], nrows=1, label="Endpoint rule")

a, b = map(QQ, interval)
t = var('t')
func = fast_callable(f(x=t), RDF, vars=[t])
dx = ZZ(b-a)/ZZ(number_of_subdivisions)

xs = []
ys = []
for q in range(number_of_subdivisions):
    if endpoint_rule == 'Left':
        xs.append(q*dx + a)
    elif endpoint_rule == 'Midpoint':
        xs.append(q*dx + a + dx/2)
    elif endpoint_rule == 'Right':
        xs.append(q*dx + a + dx)
    elif endpoint_rule == 'Upper':
        x = find_local_maximum(func, q*dx + a, q*dx + dx + a)[1]
        xs.append(x)
    elif endpoint_rule == 'Lower':
        x = find_local_minimum(func, q*dx + a, q*dx + dx + a)[1]
        xs.append(x)
ys = [ func(x) for x in xs ]

rects = Graphics()
for q in range(number_of_subdivisions):
    xm = q*dx + dx/2 + a
    x = xs[q]
    y = ys[q]
    rects += line([[xm-dx/2,0],[xm-dx/2,y],[xm+dx/2,y],[xm+dx/2,0]], rgbcolor = (1,0,0))
    rects += point((x, y), rgbcolor = (1,0,0))
min_y = min(0, find_local_minimum(func,a,b)[0])

```

```

max_y = max(0, find_local_maximum(func,a,b)[0])

pretty_print(html('<h3>Numerical integral with the {} rule</h3>'.format(endpoint_rule)))
show(plot(func,a,b) + rects, xmin = a, xmax = b, ymin = min_y, ymax = max_y)

def cap(x):
    # print only a few digits of precision
    if x < 1e-4:
        return 0
    return RealField(20)(x)
sum_html = "%s \cdot \left[ %s \right]" % (dx, ' + '.join(['f(%s)' % cap(i) for i in ys]))
num_html = "%s \cdot \left[ %s \right]" % (dx, ' + '.join([str(cap(i)) for i in ys]))

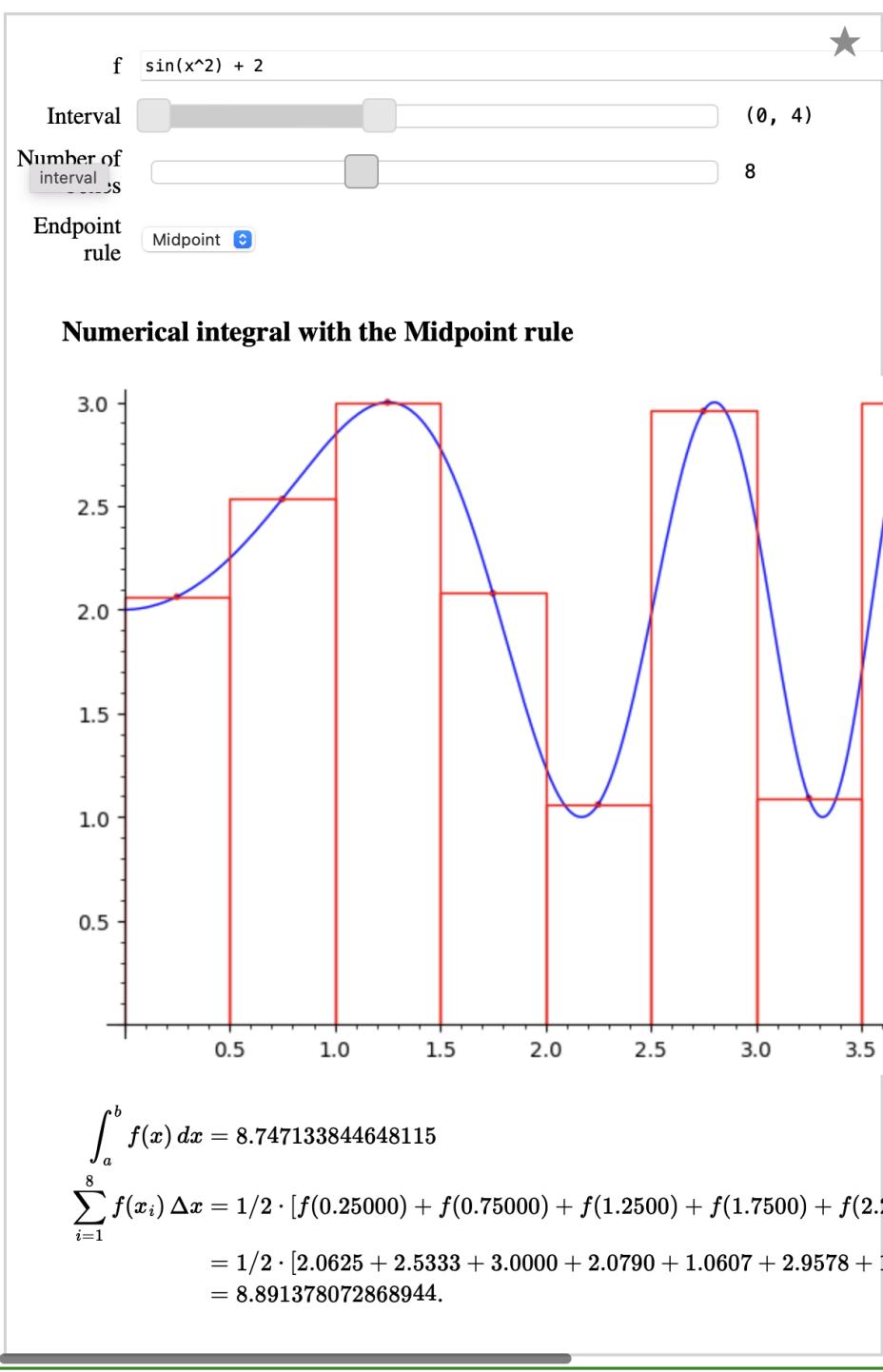
numerical_answer = integral_numerical(func,a,b,max_points = 200)[0]
estimated_answer = dx * sum([ys[q] for q in range(number_of_subdivisions)])

pretty_print(html(r'''
<div class="math">
\begin{aligned}
\int_{a}^{b} f(x) \, dx &= %s \\
\sum_{i=1}^{\infty} f(x_i) \Delta x &= %s \\
&= %s \\
&= %s . \end{aligned}
</div>'')
    % (numerical_answer, number_of_subdivisions, sum_html, num_html, estimated_answer)

```

The result of pushing Evaluate is shown on the next page.

Note that several items are interactive. A user can choose a different function at the top of the page. The number of rectangles can be varied by a slider. The height of the rectangles can be determined by the value of f on the left side, the right side, the midpoint, or by its maximum or minimum. Since Sage can integrate symbolically, the exact value of the integral is given at the bottom, followed by the Riemann Sum approximation and its value. So a wealth of examples could be suggested for students based on this one example.



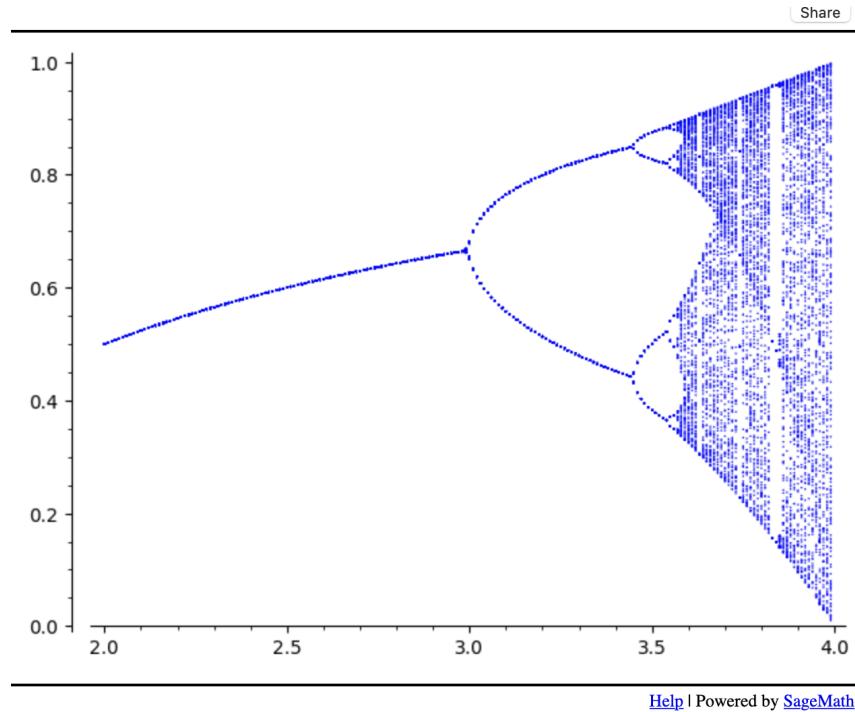
[Help](#) | Powered by [SageMath](#)

Another example contains code to calculate the Feigenbaum diagram, a canonical example in chaos theory. Many of us have probably written this code in one language or another. Here it is written in Sage.

```
N=200
M1=200
M2=200
x0=0.509434

puntos=[]
for t in range(N):
    mu=2.0+2.0*t/N
    x=x0
    for i in range(M1):
        x=mu*x*(1-x)
    for i in range(M2):
        x=mu*x*(1-x)
    puntos.append((mu,x))
point(puntos,pointsize=1)
```

Here is the result:



Other Sage examples are shown on Page 1. I'll let readers explore them.

Leaving Sage behind, the final item on Page 1 is a video from UTube. Did you know that if you right click on a UTube video, a contextual menu appears allowing you to copy code which can be pasted in a web page? The code provides that video for viewers of your page.

I found a lecture by John Maynard, one of the four Field's Prize winners at the International Congress of Mathematicians for 2022. It is fun to watch this video for the depth and clarity of his mathematics. I confess that I also watched because Maynard is left-handed and we lefties need to stick together.

Here's the code. I don't understand a word of it.

```
<iframe width="928" height="522" src="https://www.youtube.com/embed/kQqBeuk_xQw" title="13. Large gaps between primes in subsets - James Maynard (University of Oxford) [2017]" frameborder="0" allow="accelerometer; autoplay; clipboard-write; encrypted-media; gyroscope; picture-in-picture" allowfullscreen></iframe>
```

Here's a frame of the movie.



13 Creating an Interactive Page

All this is well and good, but suppose we want to create a new interactive page from scratch. How do we proceed?

As usual in TeXShop, select “New” in the File menu. A blank source page appears. Go to the “Templates” toolbar and select the template “TeX4ht-Interactive”. Some standard html boilerplate appears and then a comment reading “Insert html code here.” Erase the comment and start typing html code.

Let’s make this simple for our first example. Type

```
<p> I make this <b> bold assertion </b>. I do not believe that  
$$1 + 2 + 3 + \ldots = - \{1 \over 12\}$$ </p>
```

Notice that the template has a comment at the top telling TeXShop to typeset with the html engine. So press command-T to typeset.

Since the new source has not been saved, a save dialog appears. Navigate to the Fourier source, and to the Interactive folder in this source. Then give the source an appropriate name, like Page4. Before pushing the Save button, notice that the Save dialog has a pull-down menu named “File Format” and by default the format is “TeX File”. Change this to “HTML File” and then push Save.

Immediately the file is saved, and just as immediately it is typeset. A web page will appear showing the bold assertion and the mathematical equation we typed at the end.

After this, Page 4 can be created in the standard way. Type additional html source, and typeset to preview the result. Typesetting doesn’t actually typeset — it just opens the html file in our web view.

To link up this document with Fourier, just add a line to the Fourier source code, perhaps the code in section 9, reading

```
To read more content, click on \url{Interactive/Page4.html}.
```

Try this. Typeset Fourier again and notice that your new page is active in the document.

14 About the Template “TeX4ht-Interactive”

The head portion of the template contains the following lines:

```
<script type="text/javascript" id="MathJax-script" async  
src="https://cdn.jsdelivr.net/npm/mathjax@3/es5/tex-chtml.js">  
</script>
```

These are the lines that activate MathJax for this page.

The head portion of the template also contains the following lines:

```
<script src="https://sagecell.sagemath.org/static/embedded_sagecell.js"></script>  
<script>  
    // Make the div with id 'mycell' a Sage cell  
    sagecell.makeSagecell({inputLocation: '#mycell',  
                          template: sagecell.templates.minimal,  
                          evalButtonText: 'Activate'});  
    // Make *any* div with class 'compute' a Sage cell  
    sagecell.makeSagecell({inputLocation: 'div.compute',  
                          evalButtonText: 'Evaluate'});  
</script>
```

These are the lines that activate Sage experiments for this page.

15 Additional Interactive Elements for HTML

For serious work, additional interactive html elements are needed. For instance, a common ploy in writing interactive text for students is to write a short set of multiple choice questions. If the student answers correctly, they are encouraged to continue with the next section. If not, a dialog appears explaining why their choice was wrong and suggesting that they review a certain section of the text.

I thought about postponing this document until I had collected html code for this and similar interactions. But I'm not an html expert, and even if I found code that worked, it might be a notorious kludge for a task with a known beautiful solution.

Therefore instead of more samples, TeXShop has an expandable help file which you and other users can improve. Select the item "HTML Commands" in TeXShop Help. The result is shown on the next page.

```

3 -->
4 <!DOCTYPE html>
5 <html lang='en-US' xml:lang='en-US'>
6
7 <head>
8 <meta charset='utf-8' />
9 <meta content='width=device-width,initial-scale=1' name='viewport' />
10 <link href='Sample.css' rel='stylesheet' type='text/css' />
11 <script type="text/javascript" id="MathJax-script" async
12   src="https://cdn.jsdelivr.net/npm/mathjax@3/es5/tex-chtml.js">
13 </script>
14 <script src="https://sagecell.sagemath.org/static/embedded_sagecell.js"></script>
15 <script>
16 // Make the div with id 'mycell' a Sage cell
17 sagecell.makeSagecell({inputLocation: '#mycell',
18   template: sagecell.templates.minimal,
19   evalButtonText: 'Activate'});
20 // Make *any* div with class 'compute' a Sage cell
21 sagecell.makeSagecell({inputLocation: 'div.compute',
22   evalButtonText: 'Evaluate'});
23 </script>
24
25 </head>
26 <body>
27
28 <P><h4>Bold and Italics</h4></P>
29 |
30 <P><h5> "b" and "i" <h5></h4></P>
31
32 <p> It is <b>crucial</b> that we follow the guidelines in the <i> Style Manual</i></p>
33
34 <P><h4>Inserting a Non-Breaking Space</h4></P>
35
36 <P><h5> "nbsp" <h5></h4></P>
37
38 <p> &nbsp &nbsp &nbsp &nbsp We indent this sentence.</p>
39

```

This has got to be the ugliest help document ever designed, but that's because it contains source code. TeXShop can typeset this page, giving

Help.html

< > URL: 🔍

Bold and Italics

"**b**" and "*i*"

It is **crucial** that we follow the guidelines in the *Style Manual*

Inserting a Non-Breaking Space

"nbsp"

We indent this sentence.

Verbatim Text

"pre"

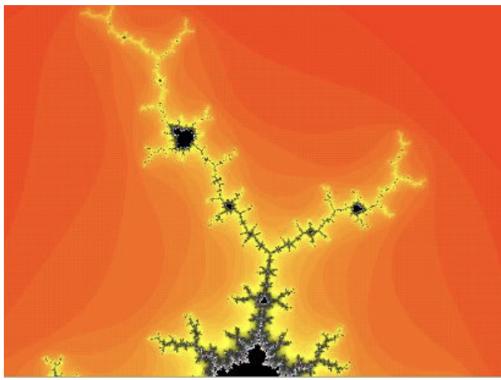
A list in LaTeX is formed with the following code:

```
\begin{itemize}
\item The sky is blue.
\item The grass is green.
\end{itemize}
```

Inserting Images

"img"

Here is a dot near the top of the Mandelbrot set:



Linking to Pdf Documents

"A" and "HREF"

Here are brief notes on the Cayley-Dickson construction and the Octonions: [Octonions.pdf](#)

Here is a set of slides at my web site: [Slides: Dirac's Belt Trick, Quaternions, and the iPad](#)

Now we have a readable help page. The advantage of this scheme is that you can easily add items to the help page. When you find a new example, add a three part item which lists its purpose, it's tags, and a short example of its use. Typeset to make sure your new example is clear. Keep adding examples as you find useful snippets of html code.

The source for this help document is the file `~/Library/TeXShop/HTML/Help.html`. You can also store support files for new items in this folder. Note that the folder already contains a movie, a pdf file, and a jpg file. Your changes are automatically saved in `Help.html`.

It is easy to share such help documents. Just zip up the folder `~/Library/TeXShop/HTML` and send the zip file to others. Feel free to send additions to me.

In this way, we may be able to collate a useful set of interactive elements for users in the future.

16 Putting These Documents on the Web

The file `Fourier.pdf` is a self-contained document, so it is easy to put it on the web. Upload it to your web site, and add a link to it from an appropriate page.

The TeX4ht project is slightly trickier to put on the web because it is not entirely contained in `Fourier.html`. The illustrations must be uploaded separately, and the `Interactive` folder must be uploaded separately. In both cases it is easiest to zip the folders, upload the zip file, and unzip on the server. Finally typesetting creates the file `Fourier.css`, which is part of the output and must be uploaded to the web. So in summary, the following objects must be uploaded and must live in the same folder on the web server:

- `Fourier.html`
- `Fourier.css`
- The entire folder `Graphics`
- The entire folder `Interactive`.

17 The Mathematical Moral

Every story should have a moral. This demonstration has two morals, one for mathematicians and one for general users.

There are different ways to teach mathematics. Some mathematicians like to give polished lectures, with every item in the correct place and every proof as elegant as possible. When I was a graduate student, a fellow student was asked in an oral exam “Tell me everything you know about finite fields. And say it in the right order, because you are going to have

to prove it in the order in which you say it.” Emil Artin was famous for giving polished lectures.

Other mathematicians like to show students how mathematics is actually created. When I had my first job, a faculty member never prepared graduate lectures. Instead he arrived in class, stated the theorem of the day, and thought out loud as he tried to find a proof strategy. Students told me that he succeeded about 75% of the time.

Still other mathematicians like to make students part of the dialog. “What should we do now?”, they ask. And when a student gives bad advice, they say “oh, that won’t work because of such and such”, but in an extremely friendly manner, so it feels like the subject is being invented on the spot.

A few mathematicians give courses on their current research, and do not know what the next lecture will be about.

Which strategy is correct? They all are correct. The wonderful thing about a University is that you have contact with many different ways to think about mathematics, each valuable.

I like to think of the pdf document as the final polished result. And I like to think of the html document as embodying the way mathematics is actually invented, using experiments, trials, false leads, and all the rest. It is very useful to have both documents, the html to remind us of the joy of discovery, and the pdf to remind us of the logical and elegant final result.

18 The Moral for Computing

The world has thousand and thousands of smart engineers, involved in projects which will take decades to reach maturity. Some of this work is open source; some is commercial; some cannot be categorized. There’s no need to gallup after every new idea, but keep paying attention.

I saw TeX4ht in its youth and dismissed it. A few months ago, I tried it again ... and realized that something fantastic had happened. It wasn’t that Eitan had missed the boat; his foundational work was all important. But TeX4ht requires MathML, it requires MathJax, and it requires the social insights that Covid forced on the world.

I watched people write MathML by hand, with giant formulas that spread across pages and pages, and it made no sense to me. Today, twenty years later, I cannot write a single MathML tag. But somehow, mysteriously, that project became essential.

I don’t recall even hearing about jsMath. Instead I noticed that the mathematics on the web was improving. It didn’t occur to me to ask why.

When I heard about Sage, I installed it, discovered that Sage code could be added to TeX documents, and played with it for several days. I kept using Mathematica for illustrations. Fifteen years later, I tried Sage on the web and got a graph. I said "nice". Then I got a 3D graph, and when I grabbed it, the graph rotated. I said "Wow, this is unbelievable!"

In the TeX world, this lesson should be second nature. We live with the Knuth miracle, and with the work of thousands who write packages, and use their work every day. But still, when something new becomes essential it is a surprise. When XeTeX showed that any language could be typeset with LaTeX, that was a revolution.

Around 1990, LaTeX made its way to the Mac because engineers wrote an editor by hand, and then wrote a dvi viewer by hand, and then modified the compiler to output faster TeX. I am in awe of those folks.

But later, the engineers at NeXT made it possible for everyone to do such tasks, and this culture spread to Apple. I could create TeXShop because Apple handed me the API for a pre-built editor, and then handed me the API for a pre-built pdf viewer, and then handed me the API to call command line programs with two lines of code.

When the web became important, every computer manufacturer built a browser. So did Apple. But then they handed me the API for a pre-built html viewer.

Pay attention. Pay attention.