

Read Me

Richard Koch

May 22, 2025

The program **tex4ht** in TeX Live converts latex source files to html web documents. This is significant for two reasons. First, correctly written html satisfies accessibility standards, and some governments are imposing similar standards on pdf documents. Authors can use tex4ht to write standard latex documents but put an html copy on the web. Second, the covid experience taught us that interactive material is essential for students, and html documents can easily support interactive elements.

TeXShop has been expanded to deal with these two concerns. It has an extra preview window which can display html documents live on the web, with active links to other sites. It is possible to write latex source code for tex4ht with sections that only appear in the pdf version and sections that only appear in the html version. The source code for sections of the html document can be written in latex or html or both. The pdf and html documents are typeset simultaneously and both versions appear in their own preview windows.

1 Fourier-for-TeXShop

To illustrate how this works, I selected a latex document about Dirichlet's convergence theorem for Fourier series, which I wrote for students years ago. No changes were required to typeset with tex4ht. To begin the demonstration, open the folder Fourier-for-TeXShop and open Fourier.tex in TeXShop. At the top of the source is a "magic comment" telling TeXShop to typeset using pdflatex. Do so. Notice that the document has lots of illustrations and lots of equations.

2 Engine Files for tex4ht

We want to typeset the same document using tex4ht. This requires a one-time task that never needs to be done again. The Fourier-for-TeXShop folder has two engine files, TeX4ht.engine and TeX4ht-Only.engine, which teach TeXShop how to typeset with tex4ht. These files need to be placed in the folder TeXShop uses for such instructions.

Select the item “Open ~/Library/TeXShop” in TeXShop’s first menu. The Finder will display a list of folders in this location, including a folder named **Engines**. Drag copies of TeX4ht.engine and TeX4ht-Only.engine into this folder. If you are warned about overwriting existing engines, do so. Earlier engines were revised for the latest TeXShop.

After adding the engines, quit TeXShop and then restart it. This is required so TeXShop will recognize the new engines.

Engine files need to have correct permissions. The two files should automatically have these permissions. But glitches sometimes occur. If you are later warned about bad permissions, find Terminal in /Applications/Utilities, run it, and type

```
cd ~/Library/TeXShop/Engines
chmod 755 TeX4ht.engine
chmod 755 TeX4ht-Only.engine
```

3 Continuing the Demo

OK, the hard stuff is out of the way. At the top of the Fourier source code, change the word “pdfflatex” to “TeX4ht” in the magic comment line and typeset again.

After a brief pause, *two* preview windows will appear. One is titled **Fourier.pdf** and the other is titled **Fourier.html**. The first is the pdf window we already saw, and the second is a new window displaying html. Resize the first window and notice that the contents magnify or shrink. Resize the html window and notice that the contents reflow. So you are really looking at html output.

Now examine the html document. Maybe it is not quite as beautiful as the original, but it is close. The illustrations still work, and the mathematical formulas are amazingly clear. Scrolling through the source code will show nothing unusual until section 9. But before we discuss that, a little history is useful.

4 History

The tex4ht project was started by Eitan Gurari of Ohio State University around 2000. It was a hard project with thousands of details to master. Mathematical equations were particularly difficult, and originally they were handled by creating a picture of each equation and pasting it in the appropriate spot.

In 2009, Eitan scheduled a talk at the annual TeX User Group meeting about braille output for the program. but just days before the conference, Eitan died unexpectedly. The project was so important that it was taken over by Michal Hoftich, Karl Berry, and a few others. Today the principal author is Michal Hoftich from the Czech Republic. Hoftich is

extremely active. A few years ago, I complained that TeX Live Utility was broken because it was showing a new update to tex4ht every day, only to be told that there had indeed been daily updates for a long time.

The most important change since Gurari concerns those little pictures of mathematical equations. It was clear at that time that the real solution would be MathML, a mathematical markup language that could be added to html for real support of mathematics. This system now exists and tex4ht is capable of outputting equations in this language. But support of MathML by web browsers is spotty and small things seem to go wrong.

In the meantime, a different system called MathJax was created; see <https://www.mathjax.org>. This non-profit organization was supported by the American Mathematical Society and has become the most important method of providing excellent support for mathematical equations in Web documents. In particular, the TeX4ht engine in TeXShop asks TeX4ht to output mathematical formulas using MathJax.

5 The MathJax Popup

Select a mathematical formula in the html version of Fourier and click on it while holding down the control key. A dialog appears as shown below. This popup comes from MathJax rather than from TeX4ht. The popup allows authors to see the actual LaTeX code which produced the formula, and much more.

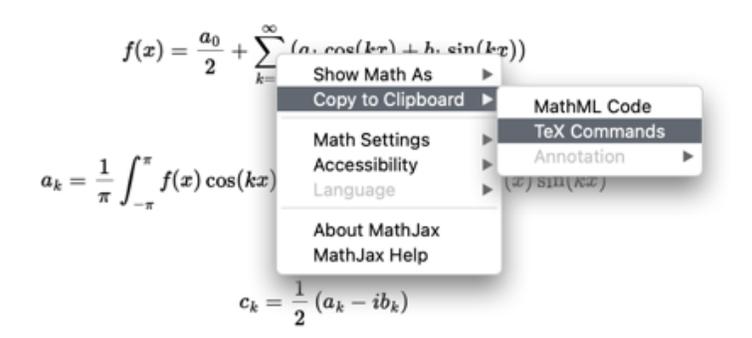


Figure 1: MathJax Popup

Although the details are not shown in the above illustration, the popup has a submenu for accessibility features. Several items are presented in this menu. One item greatly magnifies any mathematical formula when a mouse is clicked over it, and another item speaks the formula. Once set, these options are remembered and apply automatically to future MathJax sessions.

6 Fourier, Section 9

Now turn to section 9 of Fourier and compare the pdf and html output. Notice that the section titles are not the same and the contents of the two sections are dramatically different. That is because interactive content has been added to the html version of the document.

The title of the pdf section is **First Steps** and this version begins with a link to a beamer sets of slides from a talk I gave at Cal Poly Humboldt. The title of the html section is **Interactive Experiments** and the first paragraph still links to those beamer slides.

However, there is a subtle difference in those links. Go to the pdf version and click the link. Apple's pdf software displays links to offsite locations by opening the link in Safari. Now click the link in the html version. As TeXShop's html window is already active, it opens the slides without starting a secondary program. Use the back arrow key to return to the original window contents.

The source command which switches between output to pdf and output to html is

```
\ifx\HCode\undefined  
  
\else  
  
\fi
```

Here “HCode” is a variable that TeX4ht sets when it compiles a latex source file. So it is defined if TeX4ht is running and undefined otherwise. Put material for the pdf document between the first and second lines, and material for the html document between the second and third lines.

Now look carefully at the source code below this spot. We see two additional latex commands

```
\begin{html}  
  
\end{html}
```

Immediately after the first of this pair of commands, the coding changes from latex to html in the source. This html code continues for several pages until we reach the matching command that ends html coding.

The pair of commands that begin and end html encoding are latex commands, but they are only understood by TeX4ht. Our source code would fail to typeset if it encountered these commands outside an HCode–else–fi block.

After this section, the rest of the Fourier source code is pure latex.

7 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> for more information. 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. Our example project 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 web page constructed from Fourier should work for any user in the world on any operating system without installing Sage.

The syntax for calling Sage in an html program is very simple:

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

The first and last lines always remain the same, and the middle line can be any complete sage program, from a single line to an entire page of lines.

The first Sage example in Fourier is exactly the code previously shown,

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

When the html document is first viewed, a reader sees the picture below without the graph. Push **Evaluate** to see the graph. The text containing the Sage program is editable. Change this text to

```
plot(sin(x) + cos(2 * x), (x, 0, 2*pi))
```

and push **Evaluate** to try a new graph. In short, the graph in this document is truly interactive and readers can stop and try various experiments.

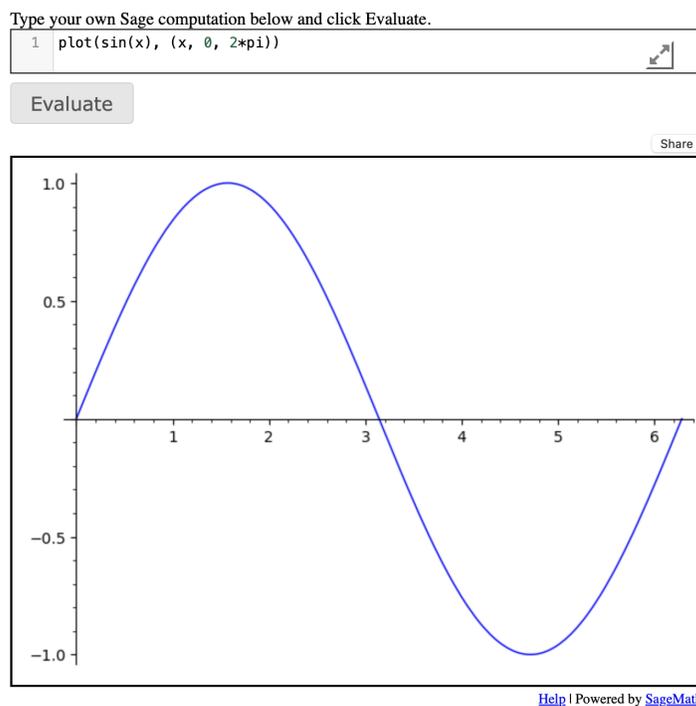


Figure 2: Sage

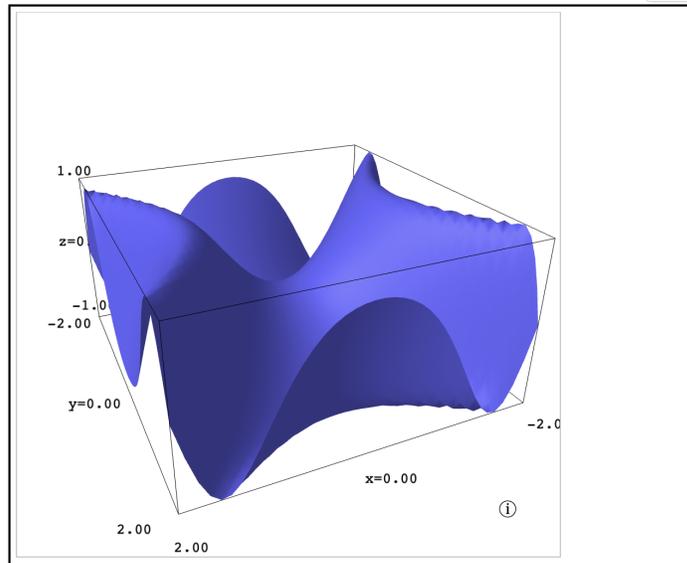
Below is another example in Fourier. The picture does not do justice to the Sage result because a user can grab the image with the mouse and rotate it, or shrink and expand it, interactively.

Type your own Sage computation below and click Evaluate.

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

Evaluate

Share



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

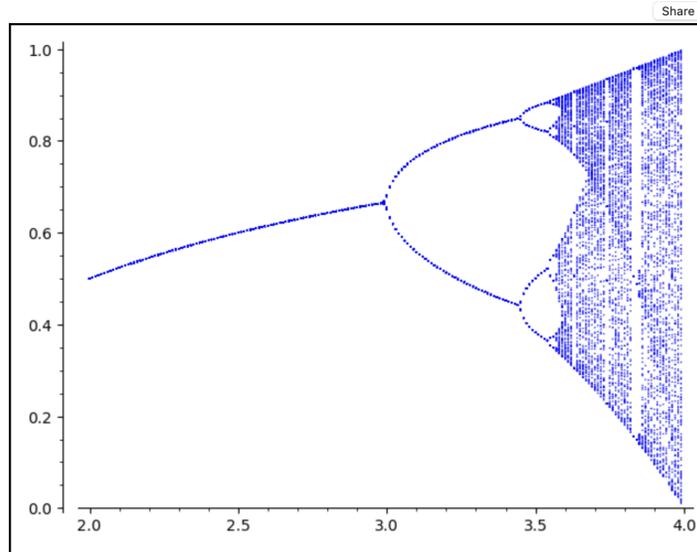
Figure 3: 3D

Sage can run entire programs. Here is a famous diagram in chaos theory, calculated in real time by the program.

Type your own Sage computation below and click Evaluate.

```
1 #Plots Feigenbaum diagram: divides the parameter interval [2,4] for mu
2 #into N steps. For each value of the parameter, iterate the discrete
3 #dynamical system  $x \rightarrow \mu x(1-x)$ , drop the first M1 points in the orbit
4 #and plot the next M2 points in a (mu,x) diagram
5
6 N=200
7 M1=200
8 M2=200
9 x0=0.509434
10
11 puntos=[]
12 for t in range(N):
```

Evaluate



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

Figure 4: Chaos Theory

The Fourier source code contains an earlier section which teaches TeX4ht how to use these simple Sage commands. The earlier section is more technical, and I'll confess that I was taught how to add it without understanding how it works. That section comes at the start of Fourier and is shown below. Add this code to the top of any source document in which you intend to call Sage.

```
\ifx\HCode\undefined
\else
\begin{html}
<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>
\end{html}
\fi
```

8 UTube Videos

If you right click on a UTube video, a contextual menu appears with code that you can copy and paste into any web page. When a user views that page, they can click on the link and run the video. As an example, I found a video of John Maynard lecturing on large gaps between primes. Maynard won a Field's Medal at the International Congress of Mathematicians in 2022. The UTube code read

```
<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>
```

I don't understand a word of this, but I copied it into the html section of the Fourier source and got a running video. The video proves that Maynard is left-handed and has a keen sense of humor. The next page shows a static sample, but the movie is active in html.



Figure 5: James Maynard

9 Mathematics in HTML

Below is a picture of the last interactive section of Fourier in the html version. At first glance this picture is unremarkable, showing a series of standard LaTeX formulas. But the code in this section is written in html rather than latex, so all of those mathematical expressions should be coded in MathML, which is painful to use.

Mathematics

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

Figure 6: Math in HTML

However, thanks to MathJax the mathematical formulas in an html section of source code can still be written in LaTeX. This is a crucial detail for authors who want to create interactive material which involves mathematical expressions. Below is the source code for the above image. Notice it is in html, but the formulas are in LaTeX.

```

456 <h3>Mathematics</h3>
457
458 <p> When  $a \neq 0$ , there are two solutions
459 to  $ax^2 + bx + c = 0$  and they are
460  $x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$ .</p>
461
462
463 Also  $\int_0^{\infty} e^{-x^2} dx = \frac{\sqrt{\pi}}{2}$ 
464
465 And  $(\alpha, \beta, \Gamma_{i_1}^{j_1 j_2})$ .
466
467 </end{html}>

```

Figure 7: Latex Inside html

There are two methods of writing mathematics in LaTeX, an old method using \$ and \$\$ and a new method using \{ and \} with curly or square brackets. Both methods work in TeX4ht for displayed formulas, but only the new methods work for inline formulas.

10 Linking to Other Web Pages

Occasionally the interactive section of a TeX4ht document may want to link to another group of independent pages. For example, the Fourier project folder contains a subfolder called /Interactive/Minimal which has six related html pages, including minimal.html which links to all the others. Section 8 of Fourier displays this example by just coding a standard link to minimal.html. The code in Fourier for this is very straightforward:

```

\ifx\HCode\undefined
  In pdf documents, there is no demo.
\else
  This is the Minimal example from the PreTeXt distribution.
  \url{Interactive/Minimal/minimal.html}
\fi

```

11 Fixing Mathematical Macros and Equation Links

Recently Greg Anderson at the University of Minnesota called my attention to an important article in Volume 70, Number 1 of the Notices of the American Mathematical Society. The article, by Eric Larson and Isabel Vogt, described two important problems that arise when converting a latex article to html, as well as giving a solution to these problems. See <https://www.ams.org/notices/202301/rnoti-p68.pdf> for the article, which contains other information not given here.

It is common to define certain textual and mathematical macros in the header of a latex document for combinations of symbols which occur often in the text. For example, the header might contain

```
\def\U{\cal U}
\newcommand{\ZZ}{\mathbb Z}
```

When tex4ht processes this source, it reads these macros. Later it applies the textual macros to the portions of text they influence. But mathematical expressions are sent to MathJax for processing, and MathJax has not seen the mathematical macros. Therefore they are not applied to the equations.

The solution to this problem is to gather all mathematical macros in a separate file called **mymacros.tex**, and send this file to MathJax before it processes equations.

The Fourier source has already been set up to illustrate this process. The source folder contains two files **mymacros.sty** and **mymacros.tex** designed for this purpose. Five mathematical macros were added to mymacros.tex. To activate the process and see how it works, go to the Fourier.tex source. In the header, find the line

```
% usepackage{mymacros}
```

and remove the comment sign. At the beginning of the first section on Waves, find the line

```
% We have  $\U$  and  $V$  and  $S$  and  $R$  and also  $\ZZ$ .
```

and remove the comment sign. Then typeset for TeX4ht, and notice that the initial line is correct in both the pdf version and the html version. That is because “the fix” was applied to correct the html version.

When you try your own project and source file, there will be more work to do. The file **mymacros.sty** need not be changed, but the file **mymacros.tex** should be emptied, and then all the mathematical macros in your header should be moved to this file. After that, add the line

```
\usepackage{mymacros}
```

to the header of your document. Then the fix will be applied automatically and html files will give correct mathematical output. Notice that textual macros work fine and should not be moved to this extra file.

The other problem is similar, but user intervention is not needed to fix it; we have provided the fix automatically. Users often add a label to equations in an equation environment. Later they want to refer to that location using **ref** or **eqref**. But the problem is that **ref** and **eqref** cannot refer to the number assigned to the equation because that was done by MathJax. This is fixed.

12 What Is MyConfig?

Inside the Fourier-for-TeXShop folder there is a subfolder named **MyConfig**. What is the purpose of this folder?

TeX4ht works in the following way to create an html file. First it typesets the source with pdflatex to create a dvi file. Then it reads the dvi file line by line and converts the various symbols to html. But latex is always changing, and new things appear in the dvi file. TeX4ht handles this problem using configuration files, which teach the program about new dvi features and how to process them.

Let me confess that I do not understand configuration files. All of the items in MyConfig are present because someone else told me to add them. The configuration file is read when TeX4ht first starts up. In the Fourier demo, we provide a configuration file in the folder MyConfig. This configuration is suitable for all of your TeX4ht projects. It contains the fixes discussed in the previous section, and also a recent fix to improve the handling of illustrations.

The engine file for TeX4ht is very simple. TeX4ht is called in the last line; notice that it is given the configuration file and the source file “\$1” and told to use mathjax.

```
#!/bin/tcsh
# !TEX-bothPreview

set path= ($path /Library/TeX/texbin /usr/texbin /usr/local/bin)
pdflatex -file-line-error -synctex=1 "$1"
make4ht -c MyConfig/myconfigfile.cfg "$1" "mathjax"
```

Aside 1: This engine calls pdflatex to typeset LaTeX, and then calls make4ht to typeset html. TeX4ht does this typesetting by first calling pdflatex to create a dvi file, and then converting this dvi file piece by piece to html.

In the TeX world, there has been recent discussion about new ways to typeset latex. Although pdflatex has been used for some time, it is being replaced by XeLaTeX and LuaLatex because these programs can process Unicode and thus languages used all over the world. Both programs are compatible with old source files; they both typeset Fourier without making any changes.

TeX4ht can easily be told to use XeLaTeX or LuaLaTeX rather than pdflatex. All that is needed is a slight revision of the previous engine file. Below is the revision for XeLaTeX; we only changed **pdflatex** to **xelatex** in the second line and added a flag “-x” in the third line. To use LuaLaTeX, use **lualatex** in the second line and change the flag to “-lua”.

```
#!/bin/tcsh
# !TEX-bothPreview

set path= ($path /Library/TeX/texbin /usr/texbin /usr/local/bin)
xelatex -file-line-error -synctex=1 "$1"
make4ht --x -c MyConfig/myconfigfile.cfg "$1" "mathjax"
```

Aside2: The second line of this engine is a fairly new addition to TeXShop. The standard behavior of TeXShop after typesetting a program like Fourier.tex is to inspect the program directory and see if it also contains a file Fourier.pdf. If so, TeXShop displays that pdf in the pdf window.

This is still the case unless the engine contains one or more of the lines below.

```
!TEX-noPreview
!TEX-pdfPreview
!TEX-htmlPreview
!TEX-bothPreview
!TEX-noConsole
```

When such a line is in the engine, it determines which window to display after typesetting ends. More details are in the TeXShop Manual, section 9.9.

By the way, TeXShop can be used to create and edit web pages. Create a file with extension **html** and open it in TeXShop. Create or edit this file in the source window using html commands. There is an engine for html in `~/Library/TeXShop/Engines/Inactive/html`. Copy this engine to the active Engines folder. Typeset your html source using this engine; your source will open in an html window and all links will be active.

13 Putting the TeX4ht Fourier Document On The Web

Suppose you have written `Fourier.tex` and it is time to put the project on the web. It is easy to put the pdf on the web because it is a single file `Fourier.pdf` and you can simply upload that file and link to it from your web page.

The TeX4ht project is trickier to put on the web because it is not entirely contained in `Fourier.html`. The illustrations must be uploaded separately and the `MyConfig` folder must also be uploaded. The configuration file inside `MyConfig` reads `mymacros.tex`, so that file must be uploaded. When uploading folders, 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`
- The entire folder `MyConfig`
- The file `mymacros.tex`

Note that during compiling, a large number of additional files are created and added to the source folder. These should not be put on the web.

14 Putting Your Own Project on the Web

Since TeX4ht adds a large number of additional files to the source folder, it pays to plan for this in advance. I strongly suggest that illustrations be placed in a separate folder, Graphics, as done in Fourier, rather than directly in the source folder. Otherwise you may have to upload a large number of files to the web. It is also crucial that mathematical macros be moved from your header to mymacros.tex. Later you will move mymacros.tex to the web. It is not necessary to move mymacros.sty to the web. Add the following line to your document header:

```
\usepackage{mymacros}
```

Suppose your project is titled Main.tex. Then you must upload

- Main.html
- Main.css
- The entire folder Graphics
- The entire folder MyConfig
- The file mymacros.tex

If you intend to place both the pdf and the html files on the web, you need to add

- Main.pdf