mathjax2mobi: Converting MathJax HTML for eBooks

This blog post was translated by ChatGPT.

Project Overview

Let's briefly discuss the project.

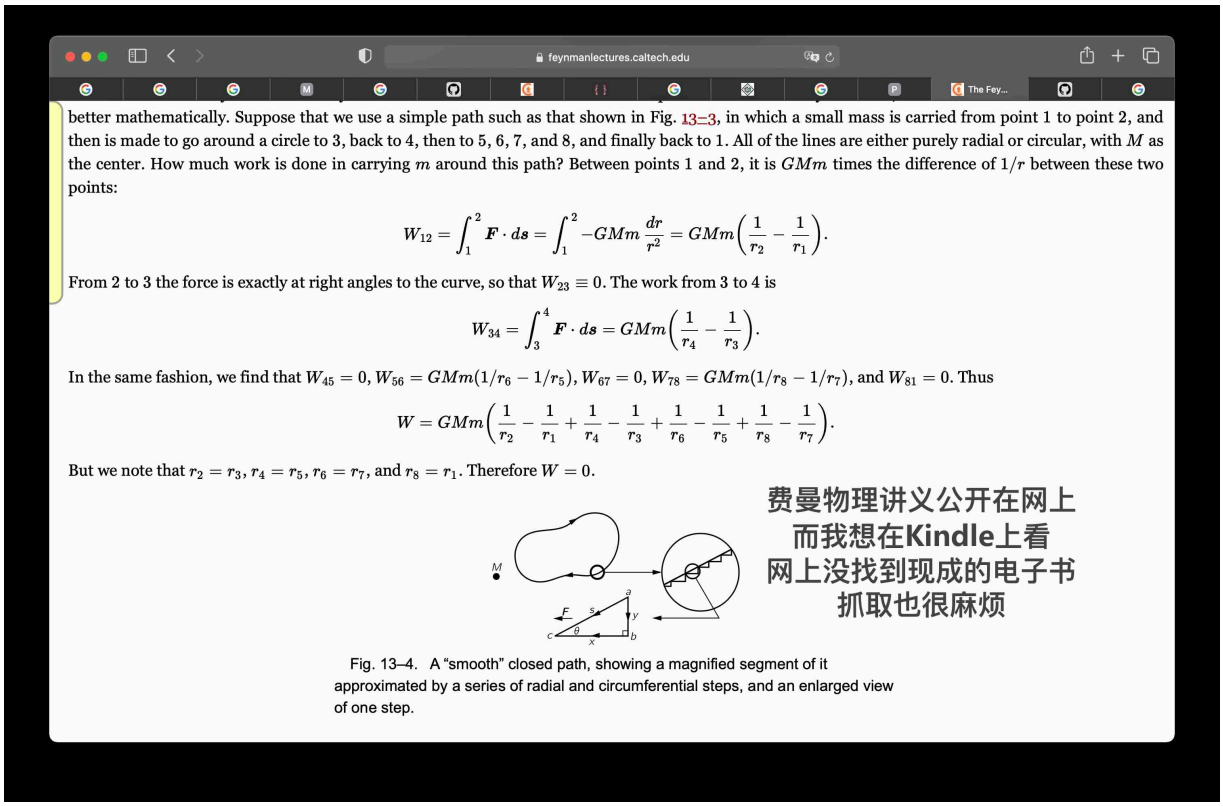


Figure 1: feynman_online

I felt a bit happy after completing the project. I wrote the following paragraph:

After coding for a day, I finally got a beautiful eBook of Feynman's Lectures on Physics! The Feynman Lectures on Physics are available online, rendered using `latex`. People often use `latex` to write papers because it renders mathematical formulas excellently. For the online version, the `mathjax` library is used. It converts `latex` source code into `html` code, generating many `div` and `span` tags. However, eBooks do not support this method. So, the idea was to scrape the webpage, reverse the `mathjax` rendering, and replace it with `svg` images. There were quite a few problems. One issue was that the source code had many custom `latex` macros that needed to be added. The second issue was that embedding many `svg` images caused problems. While a single `svg` is fine, many can cause issues, likely due to browser and `svg` bugs. To solve

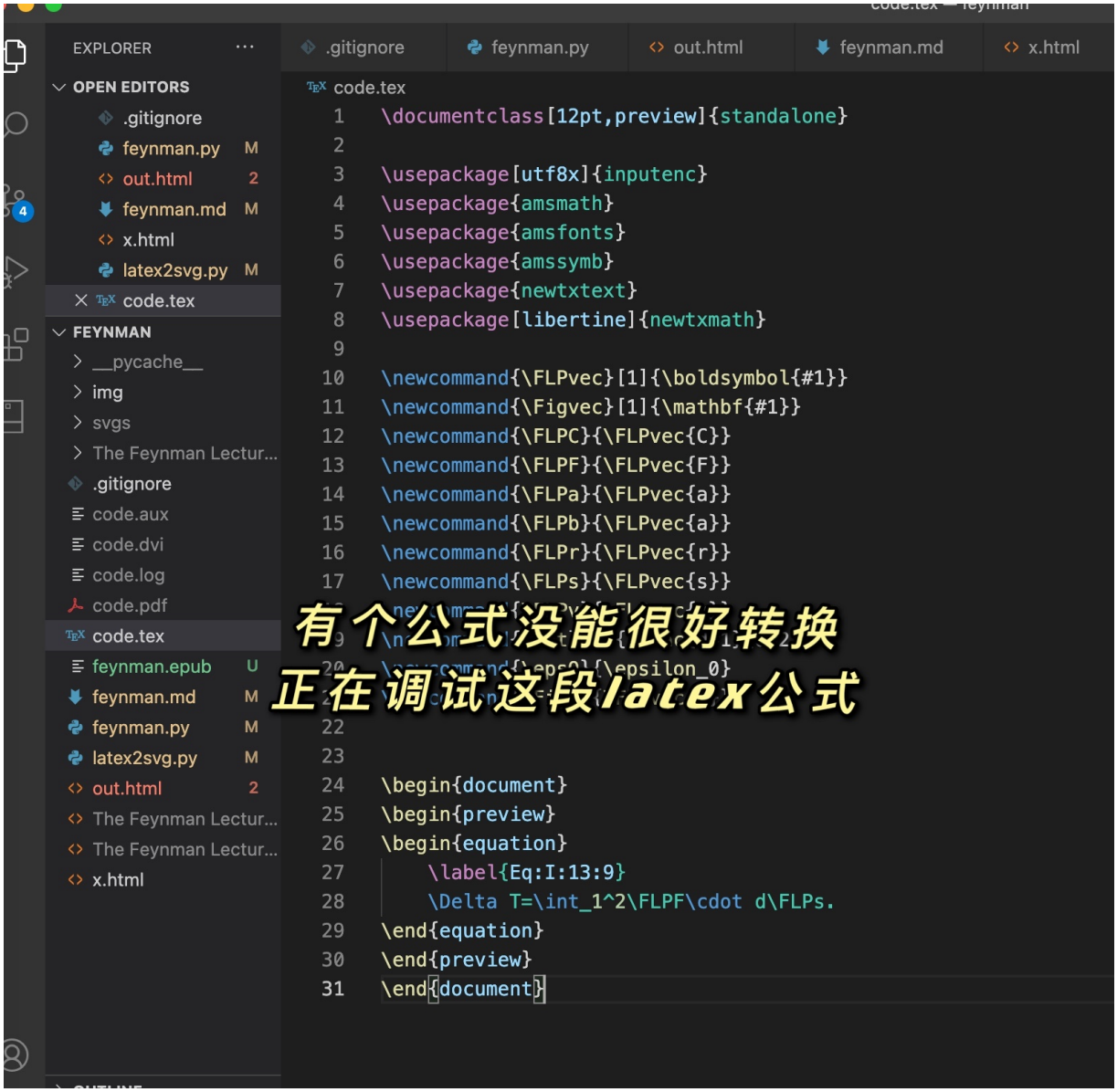


Figure 2: latex

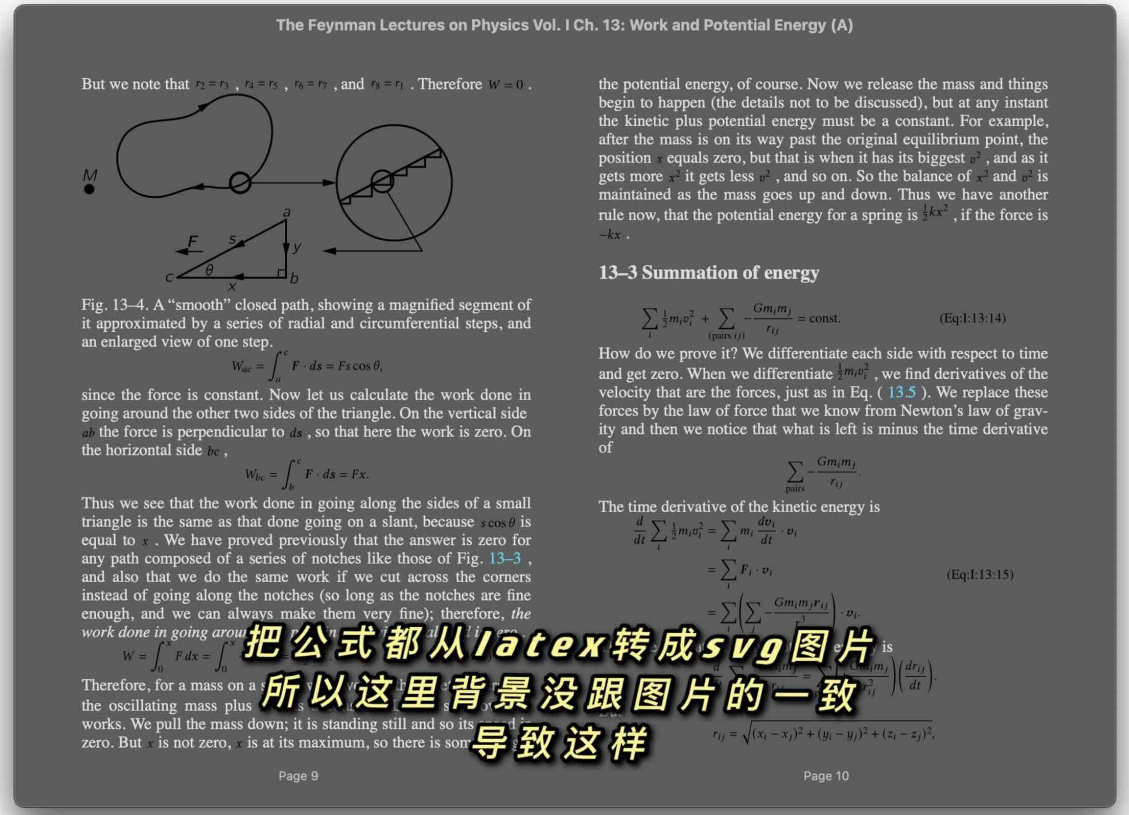


Figure 3: epub_black

From 2 to 3 the force is exactly at right angles to the curve, so that $W_{23} \equiv 0$. The work from 3 to 4 is

$$W_{34} = \int_3^4 \boldsymbol{F} \cdot d\boldsymbol{s} = GMm\left(\frac{1}{r_4} - \frac{1}{r_3}\right).$$

In the same fashion, we find that $W_{45} = 0$, $W_{56} = GMm(1/r_6 - 1/r_5)$, $W_{67} = 0$, $W_{78} = GMm(1/r_8 - 1/r_7)$, and $W_{81} = 0$. Thus

$$W = GMm\left(\frac{1}{r_2} - \frac{1}{r_1} + \frac{1}{r_4} - \frac{1}{r_3} + \frac{1}{r_6} - \frac{1}{r_5} + \frac{1}{r_8} - \frac{1}{r_7}\right).$$

But we note that $r_2 = r_3$, $r_4 = r_5$, $r_6 = r_7$, and $r_8 = r_1$. Therefore $W = 0$.

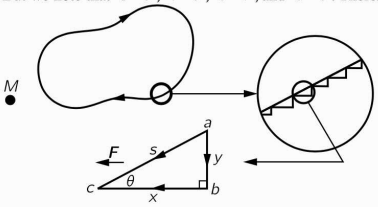


Fig. 13–4. A "smooth" closed path, showing a magnified segment of it approximated by a series of radial and circumferential steps, and an enlarged view of one step.

$$W_{ac} = \int_a^c \boldsymbol{F} \cdot d\boldsymbol{s} = Fs\cos\theta,$$

since the force is constant. Now let us calculate the work done in going around the other two sides of the triangle. On the vertical side ab the force is perpendicular to ds, so that here the work is zero. On the horizontal side bc,

$$W_{bc} = \int_b^c \boldsymbol{F} \cdot d\boldsymbol{s} = Fx.$$

Thus we see that the work done in going along the sides of a small triangle is the same as that done going on a slant, because $s\cos\theta$ is equal [illegible] We've proved previously that [illegible] any [illegible] notches like [illegible]

and also that we do the same work if we cut across the corners instead of going along the notches (so long as the notches are fine enough, and we can always make them very fine); therefore, *the work done in going around any path in a gravitational field is zero*.

$$W = \int_0^x F\,dx = \int_0^x -kx\,dx = -\tfrac{1}{2}kx^2. \qquad \text{(Eq:I:13:13)}$$

Therefore, for a mass on a spring we have that the kinetic energy of the oscillating mass plus $\frac{1}{2}kx^2$ is a constant. Let us see how this works. We pull the mass down; it is standing still and so its speed is zero. But x is not zero, x is at its maximum, so there is some energy, the potential energy, of course. Now we release the mass and things begin to happen (the details not to be discussed), but at any instant the kinetic plus potential energy must be a constant. For example, after the mass is on its way past the original equilibrium point, the position x equals zero, but that is when it has its biggest v^2, and as it gets more x^2 it gets less v^2, and so on. So the balance of x^2 and v^2 is maintained as the mass goes up and down. Thus we have another rule now, that the potential energy for a spring is $\frac{1}{2}kx^2$, if the force is $-kx$.

13–3 Summation of energy

$$\sum_i \tfrac{1}{2}m_i v_i^2 + \sum_{\text{(pairs } ij)} -\frac{Gm_i m_j}{r_{ij}} = \text{const.} \qquad \text{(Eq:I:13:14)}$$

How do we prove it? We differentiate each side with respect to time and get zero. When we differentiate $\frac{1}{2}m_i v_i^2$, we find derivatives of the velocity that are the forces, just as in Eq. (13.5). We replace these forces by the law of force that we know from Newton's law of gravity and then we notice that what is left is minus the time derivative of

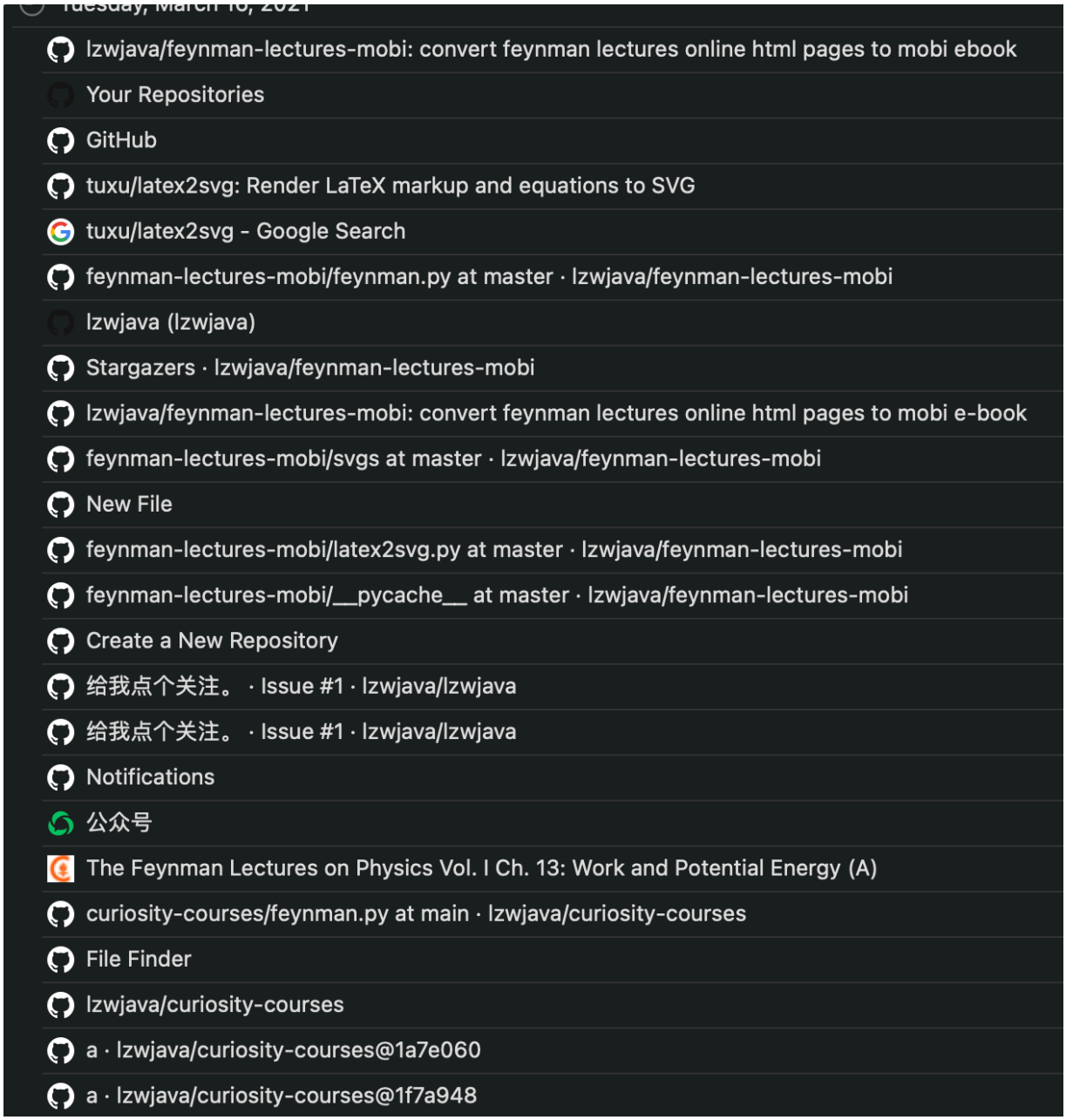
$$\sum_{\text{pairs}} -\frac{Gm_i m_j}{r_{ij}}.$$

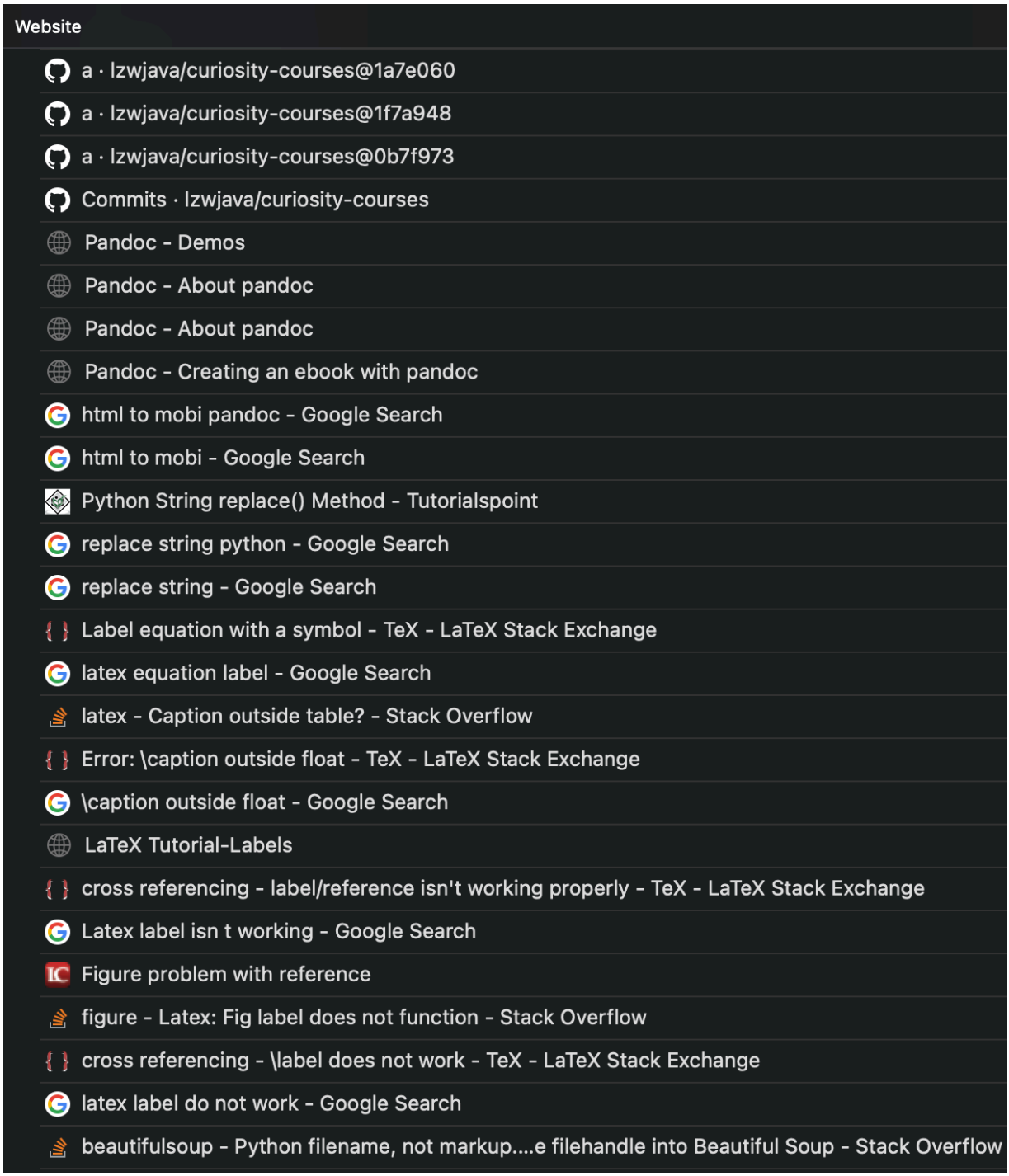
Figure 4: epub_beautiful

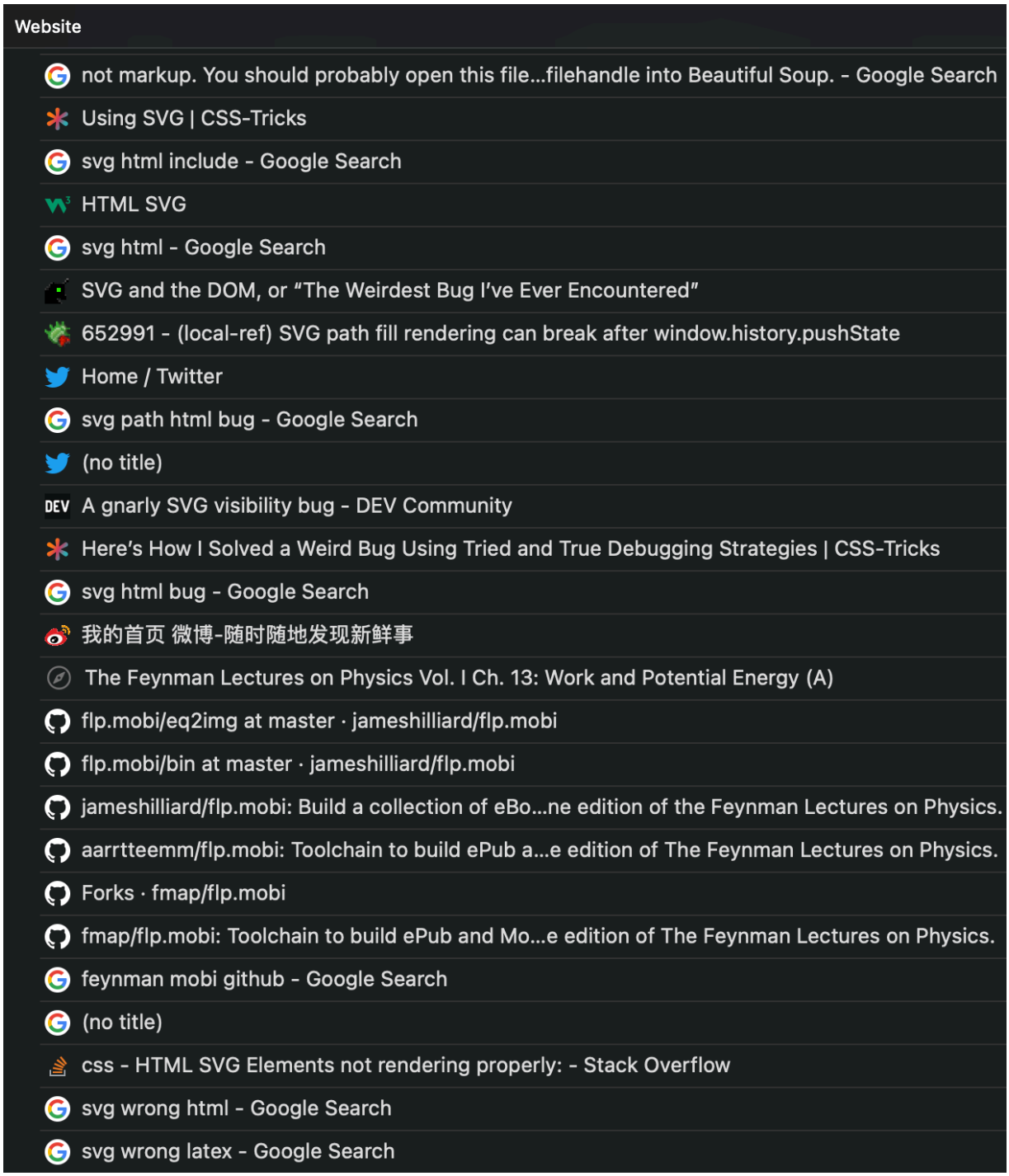
this, the `svg` images were saved as files and included with `img` tags. Formulas were also divided into two types: those in the middle of text and those that stood alone. In the end, I got a beautiful eBook!

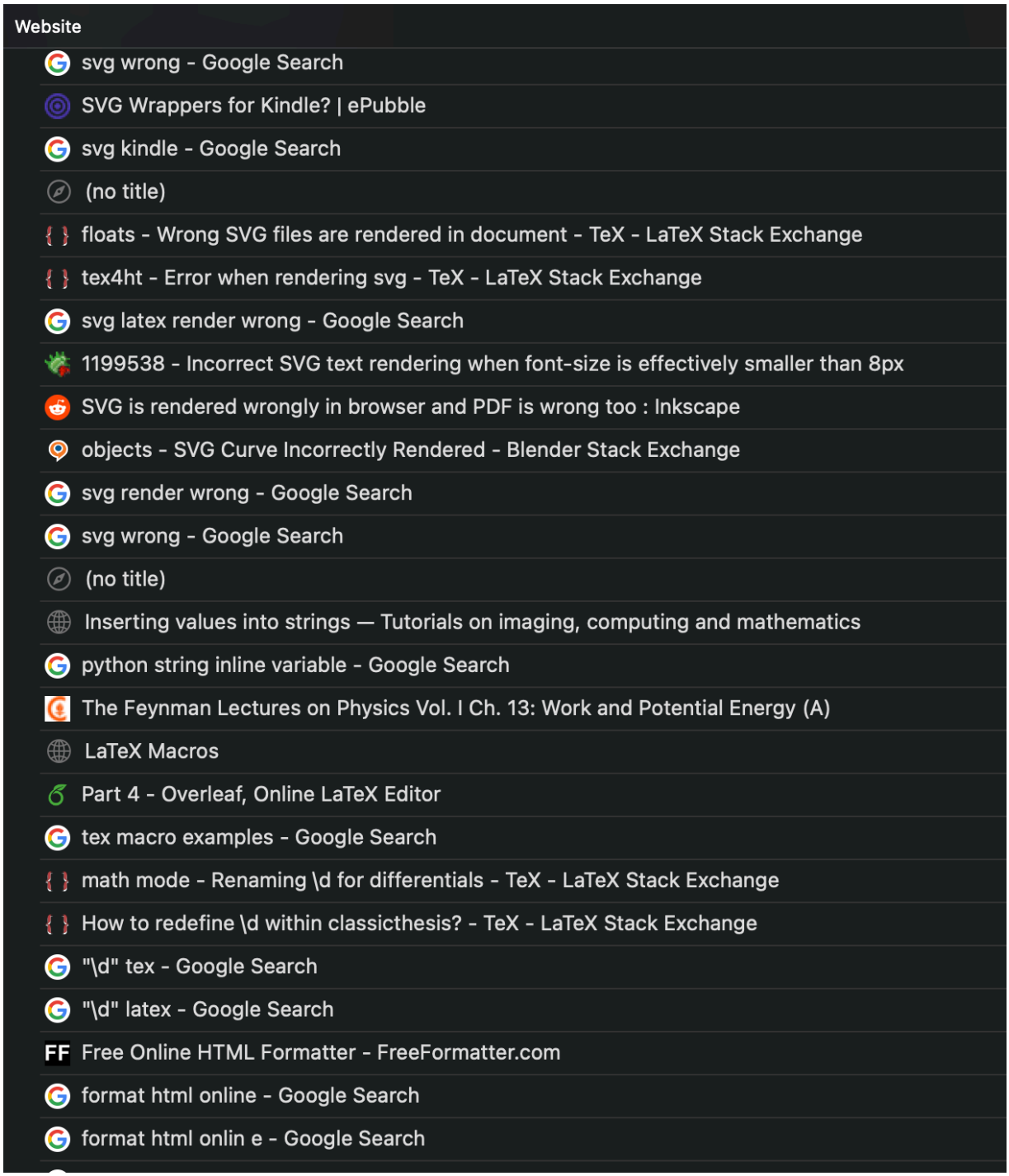
Reference Materials

Here, I document the resources accessed during the project. Since this is a tutorial, it also shows students what it's like to work on a project.



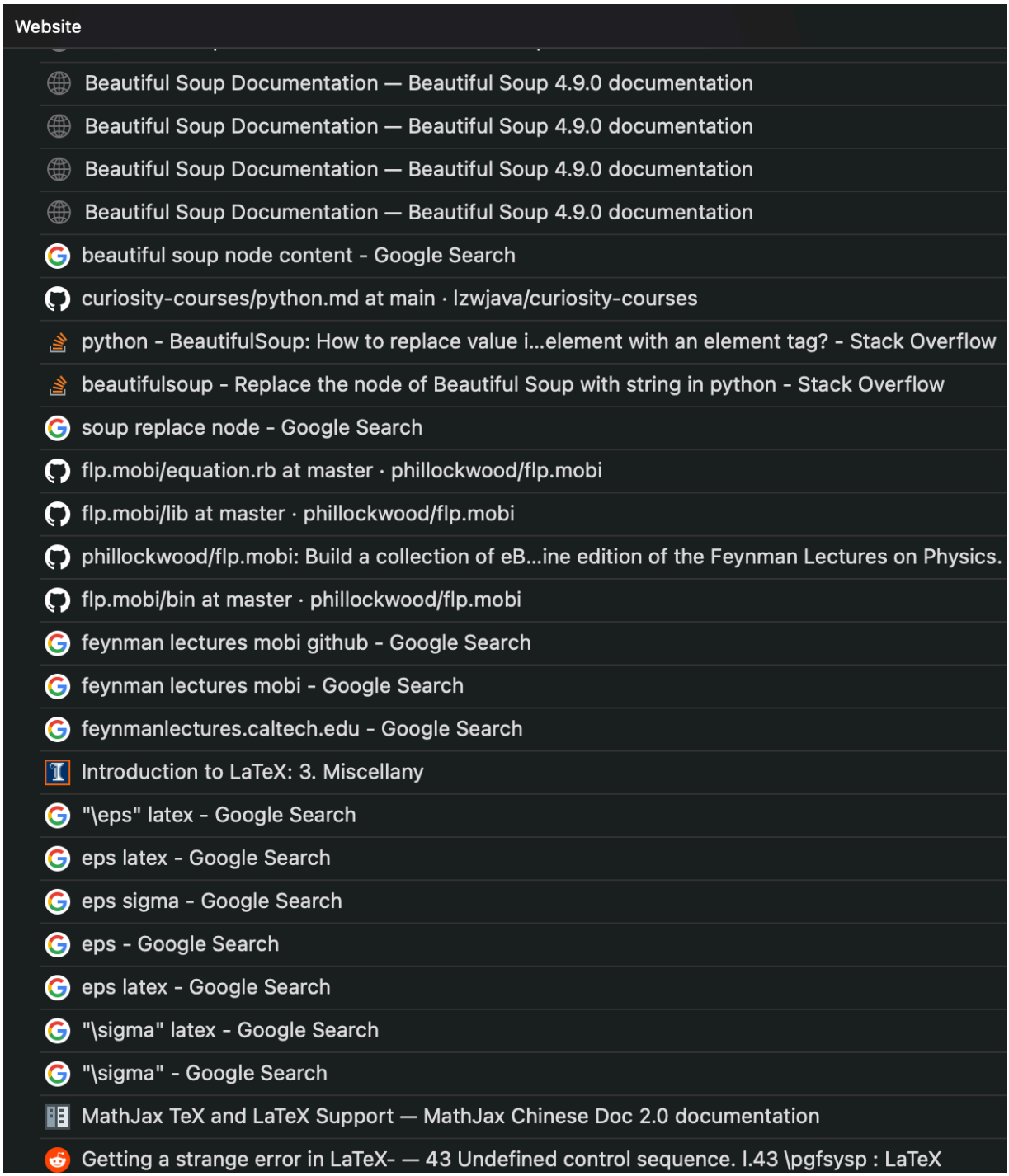

Website
a · lzwjava/curiosity-courses@1a7e060
a · lzwjava/curiosity-courses@1f7a948
a · lzwjava/curiosity-courses@0b7f973
Commits · lzwjava/curiosity-courses
Pandoc - Demos
Pandoc - About pandoc
Pandoc - About pandoc
Pandoc - Creating an ebook with pandoc
html to mobi pandoc - Google Search
html to mobi - Google Search
Python String replace() Method - Tutorialspoint
replace string python - Google Search
replace string - Google Search
Label equation with a symbol - TeX - LaTeX Stack Exchange
latex equation label - Google Search
latex - Caption outside table? - Stack Overflow
Error: \caption outside float - TeX - LaTeX Stack Exchange
\caption outside float - Google Search
LaTeX Tutorial-Labels
cross referencing - label/reference isn't working properly - TeX - LaTeX Stack Exchange
Latex label isn t working - Google Search
Figure problem with reference
figure - Latex: Fig label does not function - Stack Overflow
cross referencing - \label does not work - TeX - LaTeX Stack Exchange
latex label do not work - Google Search
beautifulsoup - Python filename, not markup....e filehandle into Beautiful Soup - Stack Overflow

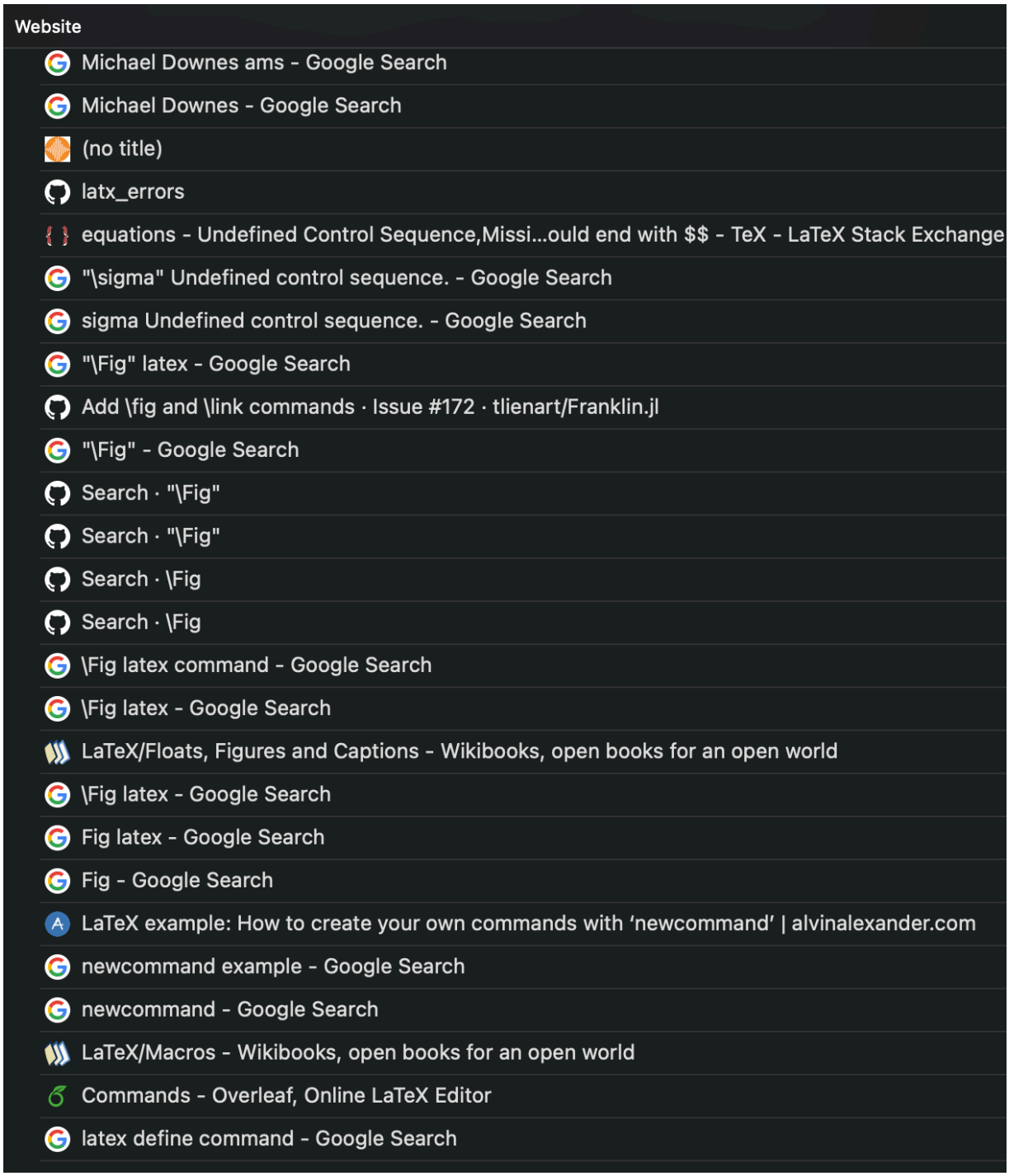

Website
not markup. You should probably open this file...filehandle into Beautiful Soup. - Google Search
Using SVG | CSS-Tricks
svg html include - Google Search
HTML SVG
svg html - Google Search
SVG and the DOM, or "The Weirdest Bug I've Ever Encountered"
652991 - (local-ref) SVG path fill rendering can break after window.history.pushState
Home / Twitter
svg path html bug - Google Search
(no title)
A gnarly SVG visibility bug - DEV Community
Here's How I Solved a Weird Bug Using Tried and True Debugging Strategies | CSS-Tricks
svg html bug - Google Search
我的首页 微博-随时随地发现新鲜事
The Feynman Lectures on Physics Vol. I Ch. 13: Work and Potential Energy (A)
flp.mobi/eq2img at master · jameshilliard/flp.mobi
flp.mobi/bin at master · jameshilliard/flp.mobi
jameshilliard/flp.mobi: Build a collection of eBo...ne edition of the Feynman Lectures on Physics.
aarrtteemm/flp.mobi: Toolchain to build ePub a...e edition of The Feynman Lectures on Physics.
Forks · fmap/flp.mobi
fmap/flp.mobi: Toolchain to build ePub and Mo...e edition of The Feynman Lectures on Physics.
feynman mobi github - Google Search
(no title)
css - HTML SVG Elements not rendering properly: - Stack Overflow
svg wrong html - Google Search
svg wrong latex - Google Search

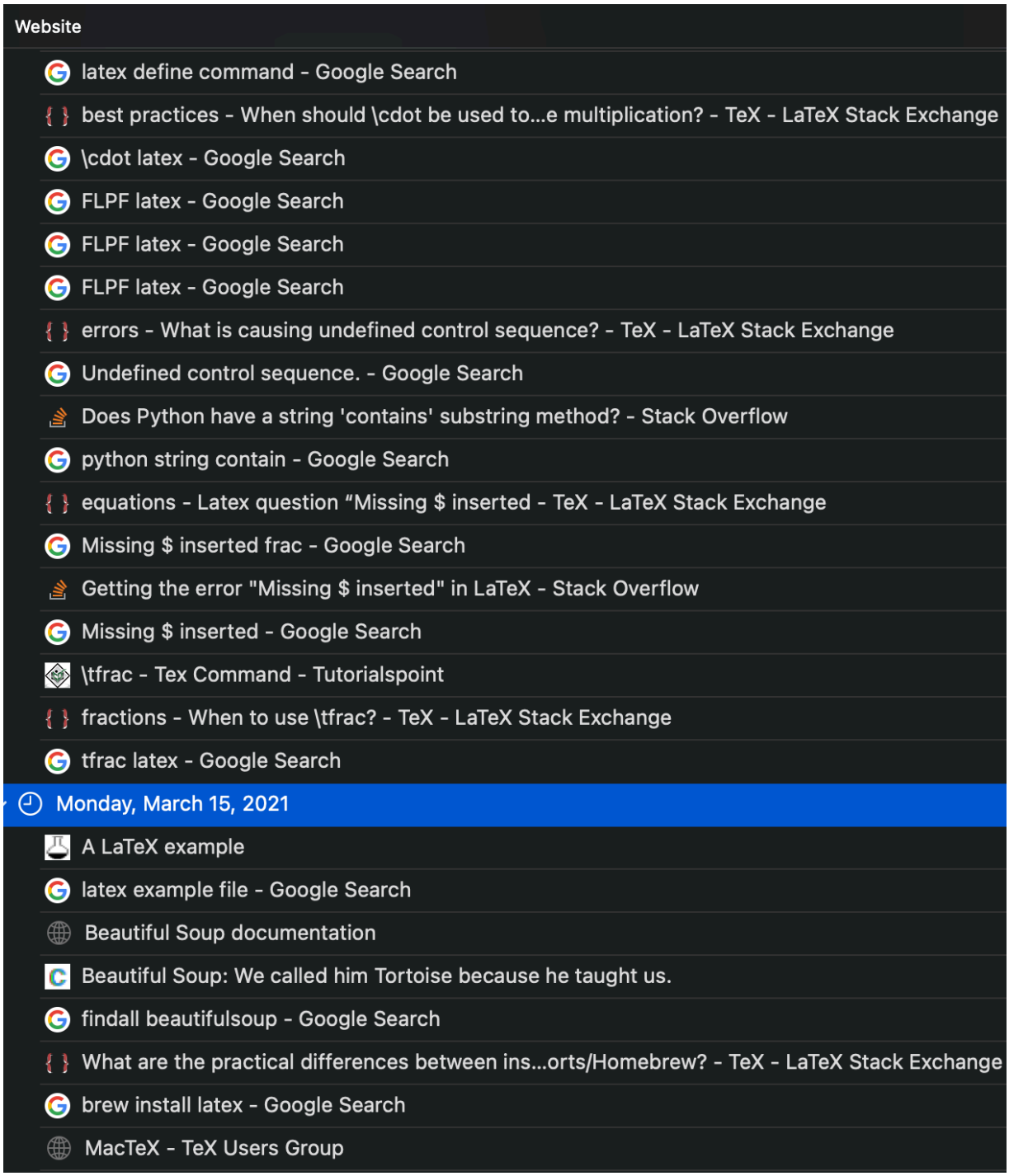

Website
svg wrong - Google Search
SVG Wrappers for Kindle? | ePubble
svg kindle - Google Search
(no title)
floats - Wrong SVG files are rendered in document - TeX - LaTeX Stack Exchange
tex4ht - Error when rendering svg - TeX - LaTeX Stack Exchange
svg latex render wrong - Google Search
1199538 - Incorrect SVG text rendering when font-size is effectively smaller than 8px
SVG is rendered wrongly in browser and PDF is wrong too : Inkscape
objects - SVG Curve Incorrectly Rendered - Blender Stack Exchange
svg render wrong - Google Search
svg wrong - Google Search
(no title)
Inserting values into strings — Tutorials on imaging, computing and mathematics
python string inline variable - Google Search
The Feynman Lectures on Physics Vol. I Ch. 13: Work and Potential Energy (A)
LaTeX Macros
Part 4 - Overleaf, Online LaTeX Editor
tex macro examples - Google Search
math mode - Renaming \d for differentials - TeX - LaTeX Stack Exchange
How to redefine \d within classicthesis? - TeX - LaTeX Stack Exchange
"\d" tex - Google Search
"\d" latex - Google Search
Free Online HTML Formatter - FreeFormatter.com
format html online - Google Search
format html onlin e - Google Search

Website

cpython/Lib at master · python/cpython
python/cpython: The Python programming language
Python
python github - Google Search
Welcome to Python.org
Beautiful Soup Documentation — Beautiful Soup 4.9.0 documentation
Beautiful Soup Documentation — Beautiful Soup 4.9.0 documentation
beautiful soup insert - Google Search
8. Errors and Exceptions — Python 3.9.2 documentation
python try catch - Google Search
LaTeX to Image converter
latex2png - convert latex equations to images
latex equation to image - Google Search
Python bs4 - find_all multiple tags and classes - DebugCN
How to get two tags in findall using BeautifulSoup | Edureka Community
Python BeautifulSoup give multiple tags to findAll - Stack Overflow
beautifulsoup find_all multiple tags - Google Search
Beautiful Soup documentation
Beautiful Soup documentation
findall beautifulsoup - Google Search
findAll - Google Search
Python Random randint() Method
python rand int - Google Search
python - How do I insert an attribute using BeautifulSoup? - Stack Overflow
python - BeautifulSoup - adding attribute to tag - Stack Overflow
beautifulsoup add attribute - Google Search


Website
Beautiful Soup Documentation — Beautiful Soup 4.9.0 documentation
Beautiful Soup Documentation — Beautiful Soup 4.9.0 documentation
Beautiful Soup Documentation — Beautiful Soup 4.9.0 documentation
Beautiful Soup Documentation — Beautiful Soup 4.9.0 documentation
beautiful soup node content - Google Search
curiosity-courses/python.md at main · lzwjava/curiosity-courses
python - BeautifulSoup: How to replace value i...element with an element tag? - Stack Overflow
beautifulsoup - Replace the node of Beautiful Soup with string in python - Stack Overflow
soup replace node - Google Search
flp.mobi/equation.rb at master · phillockwood/flp.mobi
flp.mobi/lib at master · phillockwood/flp.mobi
phillockwood/flp.mobi: Build a collection of eB...ine edition of the Feynman Lectures on Physics.
flp.mobi/bin at master · phillockwood/flp.mobi
feynman lectures mobi github - Google Search
feynman lectures mobi - Google Search
feynmanlectures.caltech.edu - Google Search
Introduction to LaTeX: 3. Miscellany
"\eps" latex - Google Search
eps latex - Google Search
eps sigma - Google Search
eps - Google Search
eps latex - Google Search
"\sigma" latex - Google Search
"\sigma" - Google Search
MathJax TeX and LaTeX Support — MathJax Chinese Doc 2.0 documentation
Getting a strange error in LaTeX- — 43 Undefined control sequence. l.43 \pgfsysp : LaTeX


Website
Michael Downes ams - Google Search
Michael Downes - Google Search
(no title)
latx_errors
equations - Undefined Control Sequence,Missi...ould end with $$ - TeX - LaTeX Stack Exchange
"\sigma" Undefined control sequence. - Google Search
sigma Undefined control sequence. - Google Search
"\Fig" latex - Google Search
Add \fig and \link commands · Issue #172 · tlienart/Franklin.jl
"\Fig" - Google Search
Search · "\Fig"
Search · "\Fig"
Search · \Fig
Search · \Fig
\Fig latex command - Google Search
\Fig latex - Google Search
LaTeX/Floats, Figures and Captions - Wikibooks, open books for an open world
\Fig latex - Google Search
Fig latex - Google Search
Fig - Google Search
LaTeX example: How to create your own commands with 'newcommand' | alvinalexander.com
newcommand example - Google Search
newcommand - Google Search
LaTeX/Macros - Wikibooks, open books for an open world
Commands - Overleaf, Online LaTeX Editor
latex define command - Google Search



Starting the Project

The Feynman Lectures on Physics are publicly available online for reading. I wanted to read them on a `Kindle`. However, since they contain many mathematical formulas, the original manuscript was likely done using `latex`. The content is displayed on the web using the `mathjax` library.

Here's an example:

```
<span class="MathJax_Preview" style="color: inherit; display: none;">
</span>
<div class="MathJax_Display">
    <span class="MathJax MathJax_FullWidth" id="MathJax-Element-10-Frame" tabindex="0" style="">
              <span class="mi" id="MathJax-Span-159" style="font-family: MathJax_Math-italic;">d<span s
                </span>
    </span>
</div>
<script type="math/tex; mode=display" id="MathJax-Element-10">\begin{equation}
\label{Eq:I:13:3}
dT/dt = Fv.
\end{equation}
</script>
```

The above is a snippet of `html` code. The `latex` raw text is within the `script` tag. `mathjax` turns it into many `span` tags for display.

Our idea now is to change `mathjax`'s display method to `svg` images.

Found a project `tuxu/latex2svg` on GitHub.

```
from latex2svg import latex2svg
out = latex2svg(r'\( e^{i \pi} + 1 = 0 \)')
print(out['depth'])
print(out['svg'])
```

Tried running it, but encountered an error.

```
    raise RuntimeError('latex not found')
RuntimeError: latex not found
```

Let's check the code.

```
    # Run LaTeX and create DVI file
    try:
        ret = subprocess.run(shlex.split(params['latex_cmd']+' code.tex'),
                             stdout=subprocess.PIPE, stderr=subprocess.PIPE,
                             cwd=working_directory)
        ret.check_returncode()
    except FileNotFoundError:
```

```
        raise RuntimeError('latex not found')
```

It turns out this also relies on the `latex` command.

Let's install it.

```
brew install --cask mactex
==> Caveats
You must restart your terminal window for the installation of MacTex CLI tools to take effect.
Alternatively, Bash and Zsh users can run the command:
  eval "$(/usr/libexec/path_helper)"
==> Downloading http://mirror.ctan.org/systems/mac/mactex/mactex-20200407.pkg
==> Downloading from https://mirrors.aliyun.com/CTAN/systems/mac/mactex/mactex-20200407.pkg
######################################################################## 100.0%
All formula dependencies satisfied.
==> Installing Cask mactex
==> Running installer for mactex; your password may be necessary.
installer: Package name is MacTeX
installer: choices changes file '/private/tmp/choices20210315-4643-5884ro.xml' applied
installer: Installing at base path /
installer: The install was successful.
   mactex was successfully installed!
```

Installation successful.

```
% latex
This is pdfTeX, Version 3.14159265-2.6-1.40.21 (TeX Live 2020) (preloaded format=latex)
 restricted \write18 enabled.
**
```

```
out = latex2svg(r'\( e^{i \pi} + 1 = 0 \)')
print(out['depth'])
print(out['svg'])


svg = open('1.svg', 'w')
svg.write(out['svg'])
svg.close()
```

`svg` generation is successful.

So, let's try generating `svg` from the `latex` text obtained from `mathjax`.

```
from bs4 import BeautifulSoup
from latex2svg import latex2svg

file = open('The Feynman Lectures on Physics Vol. I Ch. 13_ Work and Potential Energy (A).html')
content = file.read()

soup = BeautifulSoup(content)

mathjaxs = soup.findAll('script', {'type': 'math/tex'})
for mathjax in mathjaxs:
    print(mathjax.string)
    out = latex2svg(mathjax.string)
    print(out['svg'])
```

Unfortunately, it threw an error.

```
    raise CalledProcessError(self.returncode, self.args, self.stdout,
subprocess.CalledProcessError: Command '['latex', '-interaction', 'nonstopmode', '-halt-on-error', 'cod
```

Which formula caused the error?

```
\tfrac{1}{2}mv^2
```

LaTeX

Let's learn some `latex`.

```
\documentclass[12pt]{article}
\usepackage{lingmacros}
\usepackage{tree-dvips}
\begin{document}

\section*{Notes for My Paper}

Don't forget to include examples of topicalization.
They look like this:

{\small
\enumsentence{Topicalization from sentential subject:\\
\shortex{7}{a John$_i$ [a & kltukl & [el &
```

```
  {\bf l-}oltoir & er & ngii$_i$ & a Mary]]}
{ & {\bf R-}clear & {\sc comp} &
  {\bf IR}.{\sc 3s}-love   & P & him & }
{John, (it's) clear that Mary loves (him).}}
}

\subsection*{How to handle topicalization}

I'll just assume a tree structure like (\ex{1}).

{\small
\enumsentence{Structure of A$'$ Projections:\\ [2ex]
\begin{tabular}[t]{cccc}
    & \node{i}{CP}\\ [2ex]
    \node{ii}{Spec} &   &\node{iii}{C$'$}\\ [2ex]
        &\node{iv}{C} & & \node{v}{SAgrP}
\end{tabular}
\nodeconnect{i}{ii}
\nodeconnect{i}{iii}
\nodeconnect{iii}{iv}
\nodeconnect{iii}{v}
}
}

\subsection*{Mood}

Mood changes when there is a topic, as well as when
there is WH-movement.  \emph{Irrealis} is the mood when
there is a non-subject topic or WH-phrase in Comp.
\emph{Realis} is the mood when there is a subject topic
or WH-phrase.

\end{document}
```

Found a sample `latex` source code online.

```
% latex code.tex
```

```
This is pdfTeX, Version 3.14159265-2.6-1.40.21 (TeX Live 2020) (

preloaded format=latex)
 restricted \write18 enabled.
entering extended mode
(./code.tex
LaTeX2e <2020-02-02> patch level 5
L3 programming layer <2020-03-06>
(/usr/local/texlive/2020/texmf-dist/tex/latex/base/article.cls
Document Class: article 2019/12/20 v1.4l Standard LaTeX document class
(/usr/local/texlive/2020/texmf-dist/tex/latex/base/size12.clo))
(/usr/local/texlive/2020/texmf-dist/tex/latex/tree-dvips/lingmacros.sty)
(/usr/local/texlive/2020/texmf-dist/tex/latex/tree-dvips/tree-dvips.sty
tree-dvips version .91 of May 16, 1995
) (/usr/local/texlive/2020/texmf-dist/tex/latex/l3backend/l3backend-dvips.def)
(./code.aux) [1] (./code.aux) )
Output written on code.dvi (1 page, 3416 bytes).
Transcript written on code.log.
```

Let's look at the source code and the rendered effect to see what we can learn.

```
\begin{document}
\end{document}
```

This wraps the document.

```
\section*{Notes for My Paper}
```

This denotes the section title.

```
\subsection*{How to handle topicalization}
```

This denotes a subsection title.

```
\shortex{7}{a John$_i$ [a & kltukl & [el &
  {\bf l-}oltoir & er & ngii$_i$ & a Mary]]}
```

You can see that `$_i$` denotes a subscript. `{\bf l-}` denotes bold text.

```
\enumsentence{Structure of A$'$ Projections:\\ [2ex]
\begin{tabular}[t]{cccc}
    & \node{i}{CP}\\ [2ex]
    \node{ii}{Spec} &   &\node{iii}{C$'$}\\ [2ex]
```

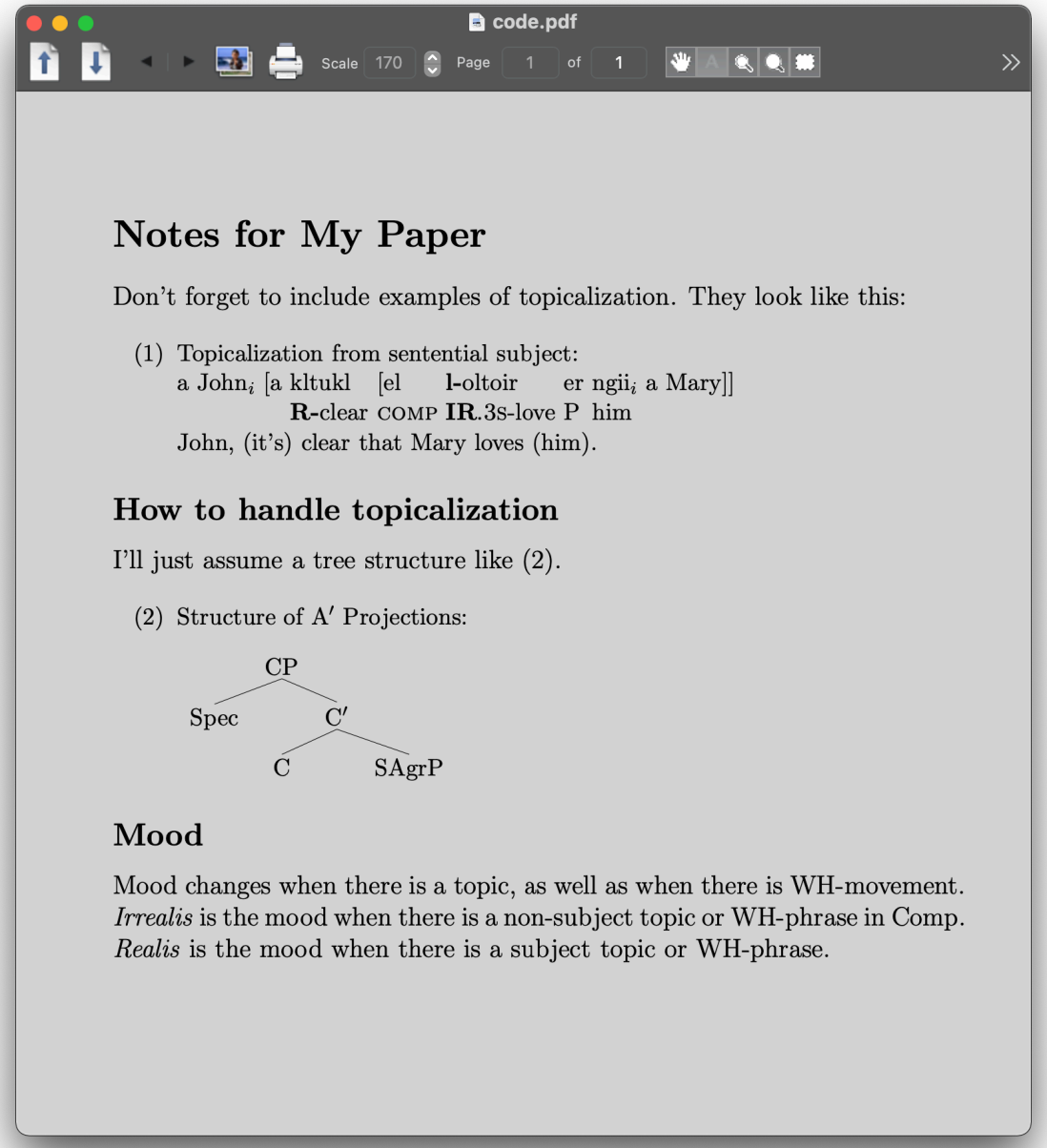


Figure 5: latex

a John$_i$ [a kltukl [el l-oltoir er ngii$_i$ a Mary]]

Figure 6: shortex

```
        &\node{iv}{C} & & \node{v}{SAgrP}
\end{tabular}
\nodeconnect{i}{ii}
\nodeconnect{i}{iii}
\nodeconnect{iii}{iv}
\nodeconnect{iii}{v}
}
```

Note that `nodeconnect` denotes connections.

Converting LaTeX to SVG

Continuing the project.

```
\documentclass[16pt]{article}
\usepackage{amsmath}
\begin{document}

\[\tfrac{1}{2}mv^2\]

\end{document}
```

This renders correctly. The failure to render in code might be due to the absence of `\usepackage{amsmath}`.

```
\documentclass[12pt,preview]{standalone}

\usepackage[utf8x]{inputenc}
\usepackage{amsmath}
\usepackage{amsfonts}
\usepackage{amssymb}
\usepackage{newtxtext}
\usepackage[libertine]{newtxmath}

\begin{document}
\begin{preview}
\tfrac{1}{2}mv^2
\end{preview}
\end{document}
```

```
! Missing $ inserted.
```

```
<inserted text>
                $
1.12 \tfrac{1}{2}
                 mv^2
```

This caused an error. Changing it as follows works:

```
\[\tfrac{1}{2}mv^2\]
```

Continuing with various attempts.

```
from bs4 import BeautifulSoup
from latex2svg import latex2svg

file = open('The Feynman Lectures on Physics Vol. I Ch. 13_ Work and Potential Energy (A).html')
content = file.read()

soup = BeautifulSoup(content, features="lxml")

mathjaxs = soup.findAll('script', {'type': 'math/tex'})
for mathjax in mathjaxs:
    print(mathjax.string)
    wrap = '$' + mathjax.string + '$'
    # if 'frac' in mathjax.string:
    #     wrap = '$' + mathjax.string + '$'
    if 'FLP' in mathjax.string:
        continue
    elif 'Fig' in mathjax.string:
        continue
    elif 'eps' in mathjax.string:
        continue
    out = latex2svg(wrap)
    # print(out)
    node = BeautifulSoup(out['svg'], features="lxml")
    svg = node.find('svg')
    mathjax.insert_after(svg)
    # print(out['svg'])
    # break
    # mathjax.replaceWith(out['svg'])
```

```
    # print(dir(mathjax))
    # break

    # out = latex2svg(wrap)
    # print(out['svg'])

# print(len(soup.contents))

output_file = open('out.html', 'w')
output_file.write(soup.prettify())
output_file.close()
# print(soup.contents)

# out = latex2svg(r'\( e^{i \pi} + 1 = 0 \)')
# print(out['depth'])
# print(out['svg'])

# svg = open('1.svg', 'w')
# svg.write(out['svg'])
# svg.close()
```

What exactly was I trying here?

```
    if 'FLP' in mathjax.string:
        continue
    elif 'Fig' in mathjax.string:
        continue
    elif 'eps' in mathjax.string:
        continue
```

When parsing the `latex` source with `FLP`, `Fig`, or `eps`, the conversion process failed.

For example, in the `HTML`, there is this script:

```
<script type="math/tex" id="MathJax-Element-11">\FLPF\cdot\FLPv</script>
```

Parsed to:

```
\FLPF\cdot\FLPv
```

When converting in code, it failed. That is, `latex2svg.py` failed. This uses the `latex` program for conversion.

`code.tex`:

```
\documentclass[12pt,preview]{standalone}

\usepackage[utf8x]{inputenc}
\usepackage{amsmath}
\usepackage{amsfonts}
\usepackage{amssymb}
\usepackage{newtxtext}
\usepackage[libertine]{newtxmath}

\begin{document}
\begin{preview}
\begin{equation}
    \FLPF\cdot\FLPv
\end{equation}
\end{preview}
\end{document}
```

```
$latex code.tex
! Undefined control sequence.
l.13     \FLPF
              \cdot\FLPv
?
```

What exactly is the problem? Later, I noticed this code in the `html`.

```
<script type="text/x-mathjax-config;executed=true">
      MathJax.Hub.Config({
        TeX: {
          Macros: {
            FLPvec: ["\\boldsymbol{#1}", 1], Figvec: ["\\mathbf{#1}", 1], FLPC: ["\\FLPvec{C}", 0], FLPI
          }
        }
      });
</script>
```

This shows that during rendering, macros were set for `MathJax`. Therefore, these macros should also be

added to our `latex` conversion source. Let' s add them.

```
\documentclass[12pt,preview]{standalone}

\usepackage[utf8x]{inputenc}
\usepackage{amsmath}
\usepackage{amsfonts}
\usepackage{amssymb}
\usepackage{newtxtext}
\usepackage[libertine]{newtxmath}

\newcommand{\FLPvec}[1]{\boldsymbol{#1}}
\newcommand{\Figvec}[1]{\mathbf{#1}}
\newcommand{\FLPC}{\FLPvec{C}}
\newcommand{\FLPF}{\FLPvec{F}}
\newcommand{\FLPa}{\FLPvec{a}}
\newcommand{\FLPb}{\FLPvec{a}}
\newcommand{\FLPr}{\FLPvec{r}}
\newcommand{\FLPs}{\FLPvec{s}}
\newcommand{\FLPv}{\FLPvec{v}}
\newcommand{\ddt}[2]{\frac{d#1}{d#2}}
\

newcommand{\eps0}{\epsilon_0}
\newcommand{\FigC}{\Figvec{C}}
\begin{document}
\begin{preview}
\begin{equation}
    \FLPF\cdot\FLPv
\end{equation}
\end{preview}
\end{document}
```

This works.

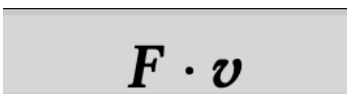


Figure 7: fv1

Analyzing the Code

Let's look at the final code.

```
import subprocess
from bs4 import BeautifulSoup
from latex2svg import latex2svg

def clean_mathjax(soup, name, cls):
    previews = soup.findAll(name, {'class': cls})
    for preview in previews:
        preview.decompose()

def clean_script(soup):
    scripts = soup.findAll('script')
    for s in scripts:
        s.decompose()

def wrap_latex(mathjax, equation = False):
    wrap = ''
    if equation:
        wrap = mathjax.string
    else:
        wrap = '$' + mathjax.string + '$'
    wrap = wrap.replace('label', 'tag')
    return wrap

def wrap_svg(svg, equation):
    if equation:
        p = BeautifulSoup(f'<div style="text-align:center;"></div>', features="lxml")
        p.div.append(svg)
        return p.div
    else:
        return svg

def to_svg(mathjaxs, equation=False):
    if equation:
```

```
            svg_prefix = 'eq_'
        else:
            svg_prefix = 'in_'
        i = 0
        for mathjax in mathjaxs:
            print(mathjax.string)
            wrap = wrap_latex(mathjax, equation=equation)
            out = {}
            try:
                out = latex2svg(wrap)
            except subprocess.CalledProcessError as err:
                raise err

            f = open(f'svgs/{svg_prefix}{i}.svg', 'w')
            f.write(out['svg'])
            f.close()

            node = BeautifulSoup('<img>', features="lxml")
            img = node.find('img')
            img.attrs['src'] = f'./svgs/{svg_prefix}{i}.svg'
            img.attrs['style'] = 'vertical-align: middle; margin: 0.5em 0;'

            p = wrap_svg(img, equation)
            mathjax.insert_after(p)
            i +=1

def main():
    file = open('The Feynman Lectures on Physics Vol. I Ch. 13_ Work and Potential Energy (A).html')
    content = file.read()

    soup = BeautifulSoup(content, features="lxml")
    clean_mathjax(soup, 'span', 'MathJax')
    clean_mathjax(soup, 'div', 'MathJax_Display')
    clean_mathjax(soup, 'span', 'MathJax_Preview')

    mathjaxs = soup.findAll('script', {'type': 'math/tex'})
```

```
    to_svg(mathjaxs, equation=False)

    mathjaxs = soup.findAll('script', {'type': 'math/tex; mode=display'})
    to_svg(mathjaxs, equation=True)

    clean_script(soup)

    output_file = open('out.html', 'w')
    output_file.write(soup.prettify())
    output_file.close()

main()
```

When we want to convert the entire eBook, we can start with a single page.

```
    file = open('The Feynman Lectures on Physics Vol. I Ch. 13_ Work and Potential Energy (A).html')
    content = file.read()
```

Here, a single page is downloaded.

`MathJax` generates many `div` and `span` tags. For example, the equation `T+U=const` generates:

```
<span class="MathJax">T</span>
<span class="MathJax">+</span>
<span class="MathJax">U</span>
<span class="MathJax">=</span>
<span class="MathJax">const</span>
```

These are annoying and affect the text because we already have `svg`. So, they are unnecessary.

```
def clean_mathjax(soup, name, cls):
    previews = soup.findAll(name, {'class': cls})
    for preview in previews:
        preview.decompose()

    clean_mathjax(soup, 'span', 'MathJax')
    clean_mathjax(soup, 'div', 'MathJax_Display')
    clean_mathjax(soup, 'span', 'MathJax_Preview')
```

Remove them all.

```
    mathjaxs = soup.findAll('script', {'type': 'math/tex'})
```

```
    to_svg(mathjaxs, equation=False)


    mathjaxs = soup.findAll('script', {'type': 'math/tex; mode=display'})
    to_svg(mathjaxs, equation=True)
```

Note that here we have two types of `script`.

```
m(dv/dt)=F
```

This is an inline formula.

```
\begin{equation}
\underset{\text{K.E.}}{\tfrac{1}{2}mv^2}+
\underset{\text{P.E.}}{\vphantom{\tfrac{1}{2}}mgh}=\text{const},\notag
```

This is a standalone formula.

For inline formulas, you need to add `$` or `[]` around the expression to avoid errors.

```
\begin{document}
\begin{preview}
\tfrac{1}{2}mv^2
\end{preview}
\end{document}
```

```
! Missing $ inserted.
<inserted text>
                $
l.26 \tfrac{1}{2}
                 mv^2
```

It should be changed to:

```
\begin{document}
\begin{preview}
$\tfrac{1}{2}mv^2$
\end{preview}
\end{document}
```

Next, let's see how to convert `latex` to `svg`.

```
    if equation:
        svg_prefix = 'eq_'
    else:
```

```
        svg_prefix = 'in_'
% tree svgs
svgs
   eq_0.svg
   eq_1.svg
   in_0.svg
```

This saves the `svg`.

```
def wrap_latex(mathjax, equation = False):
    wrap = ''
    if equation:
        wrap = mathjax.string
    else:
        wrap = '$' + mathjax.string + '$'
    wrap = wrap.replace('label', 'tag')
    return wrap
```

Here, the `latex` source is adjusted. Note that `label` is changed to `tag`.

$$\sum_i \tfrac{1}{2} m_i v_i^2 + \sum_{(\text{pairs } ij)} -\frac{G m_i m_j}{r_{ij}} = \text{const.} \qquad \text{(Eq:I:13:14)}$$

Figure 8: tag

Note the right `Eq: I: 13:14`. If it is `label`, it fails to parse correctly, displaying `(1)`. So, `tag` is used temporarily.

Next, `latex2svg.py` is called.

```
        out = {}
        try:
            out = latex2svg(wrap)
        except subprocess.CalledProcessError as err:
            raise err
```

Let's look at `latex2svg.py`.

```
    # Run LaTeX and create DVI file
    try:
        ret = subprocess.run(shlex.split(params['latex_cmd']+' code.tex'),
```

```
                            stdout=subprocess.PIPE, stderr=subprocess.PIPE,
                            cwd=working_directory)
        ret.check_returncode()
    except FileNotFoundError:
        raise RuntimeError('latex not found')
```

This calls the `latex` command.

```
 % latex --help
Usage: pdftex [OPTION]... [TEXNAME[.tex]] [COMMANDS]
   or: pdftex [OPTION]... \FIRST-LINE
   or: pdftex [OPTION]... &FMT ARGS
  Run pdfTeX on TEXNAME, usually creating TEXNAME.pdf.
```

```
    try:
        ret = subprocess.run(shlex.split(params['dvisvgm_cmd']+' code.dvi'),
                            stdout=subprocess.PIPE, stderr=subprocess.PIPE,
                            cwd=working_directory, env=env)
        ret.check_returncode()
    except FileNotFoundError:
        raise RuntimeError('dvisvgm not found')
```

This calls the `dvisvgm` command.

```
% dvisvgm
dvisvgm 2.9.1

This program converts DVI files, as created by TeX/LaTeX, as well as
EPS and PDF files to the XML-based scalable vector graphics format SVG.

Usage: dvisvgm [options] dvifile
       dvisvgm --eps [options] epsfile
       dvisvgm --pdf [options] pdffile
```

Where do we write the `latex` custom macros? Here, we need to modify `latex2svg.py` to change `default_preamble`.

```
default_preamble = r"""
\usepackage[utf8x]{inputenc}
\usepackage{amsmath}
\usepackage{amsfonts}
```

```
\usepackage{amssymb}
\usepackage{newtxtext}
\usepackage[libertine]{newtxmath}

\newcommand{\FLPvec}[1]{\boldsymbol{#1}}
\newcommand{\Figvec}[1]{\mathbf{#1}}
\newcommand{\FLPC}{\FLPvec{C}}
\newcommand{\FLPF}{\FLP

vec{F}}
\newcommand{\FLPa}{\FLPvec{a}}
\newcommand{\FLPb}{\FLPvec{a}}
\newcommand{\FLPr}{\FLPvec{r}}
\newcommand{\FLPs}{\FLPvec{s}}
\newcommand{\FLPv}{\FLPvec{v}}
\newcommand{\ddt}[2]{\frac{d#1}{d#2}}
\newcommand{\epsO}{\epsilon_0}
\newcommand{\FigC}{\Figvec{C}}
"""
```

After successful conversion, write to a file.

```
        f = open(f'svgs/{svg_prefix}{i}.svg', 'w')
        f.write(out['svg'])
        f.close()
```

Continue.

```
        node = BeautifulSoup('<img>', features="lxml")
        img = node.find('img')
        img.attrs['src'] = f'./svgs/{svg_prefix}{i}.svg'
        img.attrs['style'] = 'vertical-align: middle; margin: 0.5em 0;'
```

Here, an `img` tag is constructed.

```
def wrap_svg(svg, equation):
    if equation:
        p = BeautifulSoup(f'<div style="text-align:center;"></div>', features="lxml")
        p.div.append(svg)
        return p.div
```

```
    else:
        return svg


p = wrap_svg(img, equation)
```

For standalone `latex`, wrap it with a `div` and center it.

```
mathjax.insert_after(p)
```

Here, the `div` or `img` tag is added after the original `script`.

```
def clean_script(soup):
    scripts = soup.findAll('script')
    for s in scripts:
        s.decompose()


clean_script(soup)
```

After replacing all `latex` with `svg`, `script` tags are no longer needed, so remove them for a cleaner result.

Finally, write the modified entire `html` to a file.

```
    output_file = open('out.html', 'w')
    output_file.write(soup.prettify())
    output_file.close()
```

Then, use the `pandoc` tool to convert to `epub`.

```
pandoc -s -r html out.html -o feynman.epub
```

This will open a beautiful eBook.

Why not embed `svg` tags directly and instead use `img` tags? For example, writing it like this:

```
<p></p>
<svg></svg>
<p></p>
```

There's a very strange bug. When there are many `svg`, the following happens:

Later, I found that using `img` works. As to why, I haven't figured it out. When I took a single `svg` out and viewed it in a browser, there was no problem. It seems that when the browser renders many `svg`, it fails.

Finally

As for how to convert `epub` to `mobi`, you can use the official `Kindle` tool `Kindle Previewer 3`. Note that this is just one chapter.

The project code is available at feynman-lectures-mobi@lzwjava.

How to scrape and organize all the pages into an eBook will be discussed later. But a single chapter of the Feynman Lectures on Physics is enough for now. Alright, let' s pick up the Kindle and start reading!