DocOnce Troubleshooting and FAQ

Hans Petter Langtangen[1,2]

[1]Center for Biomedical Computing, Simula Research Laboratory
[2]Department of Informatics, University of Oslo

Jun 22, 2016

Contents

Disclaimer.

DocOnce has some support for syntax checking. If you encounter Python errors while running `doconce format`, the reason for the error is most

likely a syntax problem in your DocOnce source file. You have to track down this syntax problem yourself. However, DocOnce applies regular expressions to a large extent for transforming text, and regular expressions may sometimes lead to undesired results. Therefore, there is a chance that legal DocOnce syntax is not treated properly. Section 1.46 gives some examples of what can go wrong.

1 General topics

1.1 How can I study error messages and warnings from DocOnce?

The output from DocOnce and later compilation by various tools can be overwhelming in the terminal window. Fortunately, DocOnce also writes its warnings and error message to `mydoc.dlog` if `mydoc.do.txt` is the name of the DocOnce root file. It is recommended to study `mydoc.dlog` and correct anything that leads to warnings or errors without abortion.

1.2 How do I debug a DocOnce file that results in a strange error message?

First, do `doconce clean` to clean up all old files and recompile. Run `pdflatex` or `latex` since that checks for errors in your math formulas and is stricter on syntax than many other output formats like `html` and `sphinx`. Fortunately, DocOnce performs syntax checking on many constuctions that just lead to plain errors in LaTeX.

If you have a document consisting of a file or two, you can isolate the problem by a "bisection" method: split the file in two, compile both parts, continue with the part that gave the problem, and repeat the process. This will quickly narrow down the text so you ultimately understand the problem.

If you have a big book, it can be difficult to track down *where* the error is. If the error comes from LaTeX compilation, go to the main file's `.tex` file. Say the main document is `mydoc.do.txt`. Then find the line with the reported error message in `mydoc.tex`, find the cause of the error, *and correct the error in the corresponding DocOnce source file*. To find this file, try to recognize in `mydoc.tex` where in the document you are. Find some phrase that `X` potentially can be unique and to a Unix `find` command on `X`:

```
Terminal> find ../chapters -name '*.do.txt' -exec grep X {} \; -print
```

Hopefully, the output indentifies the right DocOnce source file.

If the error is a Mako error, you have to search in the `tmp_mako__mydoc.do.txt` file: this is the file resulting from the running the Preprocess and Mako programs, and the file is therefore the actual text that DocOnce translates.

In a big project, thre is usually some main file with a lot of `#include` statements. Go to this file, `mydoc.do.txt`, and "remove" the include statements by just removing the final `e` in include: `#includ`. Now can you can easily compile the subdocuments one by one. Start with one document file and bring in new files for inclusion until the problem appears for the first time. Then you have isolated the problematic file. It might be necessary to compile only this file. To this end, it might be necessary to insert Mako code in the beginning of the file, and specify certain command-line arguments, especially `--allow_refs_to_external_docs` so you do not get errors from missing labels that are defined elsewhere in the original big document.

Eventually, you have tracked down the erroneous piece of text, but perhaps you do not understand *why*. Then you just have to rewrite the text to something that compiles.

1.3 How do I cite and link to DocOnce?

See the "How to cite" section on the project page.

1.4 Translation aborts when labels are not defined

By default, the DocOnce translation aborts when references are found to lables not defined in the document (i.e., labels in other documents). This abortion can be annoying, and although it can be nullified by `--no_abort`, that option also ignores other error messages. Instead, use `--allow_refs_to_external_docs` to allow references to labels in other documents (these references will of course not work).

References to labels in other documents should anyway appear inside *generalized references*, see the Generalied Cross-Referencing section in the manual.

1.5 How to I control the table of contents?

To include a table of contents in the document, insert the line

```
TOC: on
```

in the DocOnce source at the place where you want the table of contents to be inserted. Often you want to do a pagebreak before the table of contents, and in that case, insert `!split` on the line before.

To control the number of section levels in the table of contents (chapters, sections, subsections), use the `--toc_depth=X` command-line option, where `X` reflects the maximum level. Level 0 correspond to chapter. The defaul value of `X`, 2, thus means including chapters, sections, and subsections. For Sphinx output, one must use the command-line option `toc_depth=X` as part of the command `doconce sphinx_dir`.

1.6 How can I add links to some resource on every page?

This question is most relevant for web documents. The simplest (and actually the only feasible) solution is to use a style that has some banner where links can be inserted.

Bootstrap HTML styles. All the Bootstrap styles have a navigation bar where one can insert custom links with the `--html_bootstrap_navbar_links=` option. Here is an example where we insert two links:

```
Terminal> doconce format html tmp1 --html_style=bootstrap \
          "--html_bootstrap_navbar_links=\
          Google|http://google.com;\
          DocOnce formats|\
          http://hplgit.github.io/teamods/writing_reports/index.html"
```

This command is broken over several lines but should be written on one line. The syntax for custom links is `link|url;link|url;link|url`. This example will result in a navigation bar with four links: a link to the document using the title as text, a link to "Google", a link to "DocOnce formats", and to the right a "Contents" pull-down menu.

Vagrant HTML template. DocOnce comes with the Vagrant HTML template, which allows for custom links in the header (here is an example on a document). Copy the `template_vagrant.html` file and the `css` directory to the directory containing the DocOnce document. Edit the `LogoWord` and `withSubWord` in the `template_vagrant.html` and insert your own links for `GO TO 1` and `GO TO 2`. Compile the document with

```
Terminal> doconce format html mydoc --html_template=template_vagrant.html
```

Sphinx bootstrap theme. With the `bootstrap` theme, Sphinx can easily offer custom links. You have to edit the generated `conf.py` in the Sphinx directory (called `sphinx-rootdir` by default). Search for `elif html_theme == 'bootstrap` in this file and edit the `navbar_links` item. Here is an example:

```
elif html_theme == 'bootstrap':
    ...
    html_theme_options = {
        'navbar_title': short_title,
        # Global TOC depth for "site" navbar tab. (Default: 1)
        # Switching to -1 shows all levels.
        'globaltoc_depth': -1,
        ...
        # A list of tuples containing pages or urls to link to.
        ...
        'navbar_links': [
           ('Google', 'http://google.com', True),
           ('DocOnce formats',
            'http://hplgit.github.io/teamods/writing_reports/index.html',
            True),
           ],
```

One may hand-edit the `conf.py` file and provide a custom version in the `doconce sphinx_dir` command:

```
Terminal> doconce sphinx_dir conf.py=../customfiles/conf.py \
          theme=alabaster toc_depth=2 mydoc
```

Alternatively, one may auto edit the generated `conf.py` file in a script:

```
doconce format sphinx mydoc
doconce sphinx_dir theme=alabaster toc_depth=2 mydoc
# Auto edit conf.py
doconce subst "#'navbar_links':.+" "'navbar_links': [
           ('Google', 'http://google.com', True),
           ('DocOnce formats',
            'http://hplgit.github.io/teamods/writing_reports/index.html',
            True),
           ]," mydoc.rst
python automake_sphinx.py
```

1.7 Why are underscores in text a potential problem?

Underscores in plain text are not tolerated by LaTeX, but works fine in other formats. In LaTeX, such underscores can give different obscure error messages, depending on the context. The remedy is to *typeset all words with underscores in inline verbatim*, e.g.,

```
If you choose the ‘Enable_access‘ menu, ...
```

The downside is that the word with underscores is typeset in monospace font.

The standard remedy in LaTeX is to insert a backslash before the underscore:

```
If you choose the Enable\_access menu, ...
```

but the backslash is not wanted in other formats. A Mako function can fix this:

```
<%
def underscorephrase(phrase):
    if FORMAT in ('pdflatex', 'latex'):
        return phrase.replace('_', '\\_')
    else:
        return phrase
%>

# Some text:
If you choose the ${underscorephrase('Enable_access')}, ...
```

1.8 GitHub gh-pages will not display files starting with a dot or underscore

This is a typical problem related to publishing either splitted HTML documents or Sphinx versions of documents on GitHub, using a gh-pages branch. Filenames starting with a dot (as in `._mydoc001.html`) or located in a directory whose name starts with an underscore (as in `spninx/_static`) will not be properly shown in HTML if not the root directory of the gh-pages branch contains a file `.nojekyll`. This file can be empty. A typical fix is

```
Terminal> git checkout gh-pages
Terminal> touch .nojekyll
Terminal> git add .nojekyll
Terminal> git commit -am updates
Terminal> git push origin gh-pages
```

1.9 GitHub gh-pages will not display files

A common error is to link to addresses starting with `https://github.com` rather than the required `http://username.github.io/`.

1.10 "Blank line is illegal in latex block", but there is no blank line

Are you compiling for `sphinx`, `ipynb`, `pandoc`? In that case, `align` environments are rewritten as sequences of plain `equation` environments (since MathJax for these output formats does not support numbered equations in `align`). If a linebreak `\\` in the `align` environment appears on its own line, this line will be blank in the equation environment and cause the error message. Simply move the linebreak up at the end of the preceding line.

Note: This kind of errors are now automatically removed (since Dec, 2015).

1.11 A newline in equations (double backslash) becomes single backslash and eats up spaces

This is a problem with the Mako preprocessor and files that have been edited on Windows. The remedy is to change the newline character on Windows to that in Unix:

```
Terminal> doconce subst '\r\n' '\n' mydoc.do.txt
```

If you work on Windows, you probably have to avoid using the Mako preprocessor (it is not used unless you use Mako functions, variables, or control statements).

1.12 Preprocess error: unexpected EOF while parsing

This error may arise from a syntax error with a colon at the end of if tests, e.g. `# #if FORMAT == 'html':`. Colon is not used with Preprocess (but required by Mako: `if FORMAT == 'html':`).

1.13 When I set a large portion of text in color, only a part of it becomes colored

Although DocOnce features a color inline tag (`color`), this tag will most likely fail for large portions of text. The reason is that this tag, like most other DocOnce constructions, relies on a regular expression for being detected. The regular expression searches for opening and closing braces after the `color{red}` specification. The closing brace is the first brace encountered. Therefore, if you have any text containing a right brace inside the text to be colored, the first right brace will define the end of the coloring. Here is an example:

```
color{blue}{Here is some text
to be typeset in blue.

===== Here is a new subsection =====
label{sec:subsec}

More text to be typeset in blue, but the right brace in the label
will end the coloring. There is no way out of this except for
moving constructions with braces (‘label‘, ‘cite‘) out of the
text to be colored.
```

1.14 How can I make use of a native LaTeX environments (example environment for instance)?

LaTeX writers often lack their favorite environments, but DocOnce has a way to define such environments, with corresponding typesetting also for HTML and all other output formats. See the section User-Defined Environments in the DocOnce manual. It contains all the details for using a standard example or theorem environment in LaTeX output.

1.15 How can I create a more tailored code environment in LaTeX than what DocOnce offers?

It is easy with *user-defined environments*, see such an example at the end of the document Demonstration of DocOnce support for LaTeX code block environments. Description of user-defined environments are found in the DocOnce manual.

1.16 How can links open in new tabs/windows

Use the option `--html_links_in_new_window` for the `doconce format` command. It works in HTML and Sphinx, but not in pandoc (or other Markdown formats) or wikis.

1.17 Movies will not play in my broswer

Are you using Safari as browser? It will not play Ogg and WebM movies. A message in a box should notify you about this (right below the movie player).

1.18 Reveal slides look strange in the browser

Are you using Safari as browser? reveal.js slides work best in Firefox, Chrome, or Opera.

1.19 How can I control the vertical spacing in slides?

HTML5 and LaTeX slides control the spacings for you. However, sometimes you really want to add some space. The `<linebreak>` is effective for this purpose. For example,

```
!bblock (large) So - how to be excellent?
<linebreak>
Excellence is not a planned goal - it is the corollary of
deep passion, very much hard (and exciting!)
work, and *thinking constantly about it*.
!eblock
```

is rendered as

So - how to be excellent?.

Excellence is not a planned goal - it is the corollary of deep passion, very much hard (and exciting!) work, and *thinking constantly about it*.

The `<linebreak>` is a newline in LaTeX if it has preceding text, otherwise it is a `\vspace{3mm}`. In HTML, `<linebreak>` is `<br>`.

More vertical space is obtained by repeated use of `<linebreak>`:

```
<linebreak>
<linebreak>
<linebreak>
```

1.20 How do I insert a copyright?

See the Copyright section of the DocOnce manual.

Other formats. An error message is provided.

1.21 How can I get a copyright on every slide page?

HTML formats: Use the `-copyright=everypage` option in the `doconce split_html` or `doconce slides_html` commands.

LaTeX Beamer: The copyright is inserted as a `\logo{}` specification, but this construction does not work in general. Then you cannot get the copyright on every page. The copyright is inserted as part of the date command so it is visible on the front page.

Recall that `{copyright...}` specification(s) must be part of the `AUTHOR:` command in order to get any copyright notice at all.

1.22 How do I get "Figure" written in another language?

DocOnce documents can be written in any language, but figure captions typically contain the English word "Figure". This must be fixed by auto-editing the output file after translation by `doconce format`.

In LaTeX you can use your native language package, e.g.,

```
doconce replace '% insert custom LaTeX commands...' '\\usepackage[norsk]{babel} mydoc.tex
```

to get the Norwegian language.

In other formats you have do more manual substitutions:

```
doconce replace Figure Figur mydoc.html
doconce replace figure figur mydoc.html
```

or both in one command using regular expressions:

```
doconce stubst '([Ff]igur)e' '\g<1>' mydoc.html
```

1.23 Spellcheck reports mistakes that I cannot find

The `doconce spellcheck` command strips off a lot of text (math, code, syntax tags) before it starts spellchecking. The spellchecked text is therefore different from the doconce source! Fortunately, the file that is actually spellchecked can be examined. If the spellcheck command is

```
doconce spellcheck mydoc
```

the file `tmp_stripped_mydoc.do.txt` contains the text that is actually run through the spellchecker (`ispell`). Search for the misspelled word there and try to find it in the doconce source code. Very often, strange misspellings come from a syntax typo that has led to a strange word in the stripped version.

Wrong typesetting of mathematics can lead to strange errors, e.g., misspellings such as `iu` because of some math `f_iu_i` that is not properly typeset between dollar signs. Here are some useful grep commands to search for such errors:

```
grep --color=auto -E '\biu\b' tmp_stripped_mydoc.do.txt
grep --color=auto -E ' iu' tmp_stripped_mydoc.do.txt
grep --color=auto -E 'iu ' tmp_stripped_mydoc.do.txt
grep --color=auto -E 'iu' tmp_stripped_mydoc.do.txt
```

1.24 Spellcheck reports a lot of mistakes related to LaTeX math

The `doconce spellcheck` command should ignore LaTeX math, but if the dollar signs for inline math are not correct (one missing, for instance), a lot of math enters the text to be spellchecked. Invoke the relevant `tmp_missing_*` file and find the first math-style expression that is reported as misspelling. Open the corresponding stripped file, `tmp_stripped_*`, which is supposed to have all the math stripped away, and search for the misspelling. When you find it, you will see that there are math expressions in the stripped file that should not be there. (Because of wrong begin and end signs around math expressions ordinary text has instead been stripped away. This way, a missing dollar sign can lead to hundreds of misspellings.) Find the problem in the corresponding DocOnce file and correct it. A similar error can be caused by wrong matching of equation environments between `!bt` and `!et`.

1.25 Text between subexercises are moved

If you insert text between a `!esubex` and the next `!bsubex`, this text is moved before all the subexercises. This is a feature, not a bug (exercises have certain elements: main text, subexercises, hints, etc. that are typeset in a specific order, which may be different from what appears in the DocOnce source file). If you need a comment between two subexercises, just place the comment at the end of the previous subexercise.

1.26 DocOnce aborts because of a syntax error that is not an error

DocOnce searches for typical syntax errors and usually aborts the execution if errors are found. However, it may happen, especially in verbatim blocks, that DocOnce reports syntax errors that are not errors. To continue execution, simply add the `--no_abort` option on the command line. You may send an email to the DocOnce author at `hpl@simula.no` and report the problem.

1.27 Figure captions are incomplete

If only the first part of a figure caption in the DocOnce file is seen in the target output format, the reason is usually that the caption occupies multiple lines in the DocOnce file. The figure caption must be written as *one line*, at the same line as the FIGURE keyword.

1.28 Problems with boldface and emphasize

Two boldface or emphasize expressions after each other are not rendered correctly. Merge them into one common expression.

1.29 Links to local directories do not work

Links of the type

```
see the "examples directory": "src/examples"
```

do not work well. You need to link to a specific HTML file:

```
see the "examples directory": "src/examples/index.html"
```

We recommend to put all files you link to in a `_static` directory if you intend to use the `sphinx` output. This guarantees that all your files are collected in the Sphinx directory tree bundle. With plain `html` output only, you can link to whatever, but remember to move all files you link to if you move the primary `.html` file.

1.30 Links are not typeset correctly

Not all formats will allow formatting of the links. Verbatim words in links are allowed if the whole link is typeset in verbatim:

```
see the directory "‘examples‘": "src/examples/index.html".
```

However, the following will not be typeset correctly:

```
see the "‘examples‘": "src/examples/index.html" directory.
```

The back-ticks must be removed, or the text can be reformulated as in the line above it.

1.31 Inline comment does not handle verbatim or emphasis right

Inline comments lead (in LaTeX output formats) to shortened titles for a table of inline comments, and the shortening is done naively: the title can cut a verbatim expression in the middle. There is no other way around that to rewrite the inline comment. (The title consists of the first 50 characters or so of the text.

1.32 Inline verbatim code or boldface is not detected when used with a footnote

There must be a space between the verbatim code (or boldface words) and the footnote bracket:

```
The construction ‘def f(x,y):‘[^whitespace] is standard.

[^whitespace]: Well, a space before the ‘y‘ argument, as in ‘f(x, y)‘,
would be the standard.
```

1.33 Inline verbatim code is not detected

Make sure there is a space before the first back-tick.

1.34 Inline verbatim text is not formatted correctly

Make sure there is whitespace surrounding the text in back-ticks.

1.35 Strange non-English characters

The former reason for this problem is that DocOnce could only work with latin1 (ISO-8859) encoding and not UTF-8. After May 2013, DocOnce applies UTF-8 both for HTML and LaTeX.

Check the encoding of the `.do.txt` file with the Unix `file` command or with

```
Terminal> doconce guess_encoding myfile.do.txt
```

If the encoding is different from ASCII or UTF-8, say latin1, convert to UTF-8 using either of the Unix commands

```
Terminal> doconce change_encoding latin1 utf-8 myfile.do.txt
Terminal> iconv -t utf-8 -f latin1 myfile.do.txt --output newfile
```

1.36 Wrong Norwegian characters

When DocOnce documents have characters not in the standard ASCII set, the format of the file must be UTF-8. See the section "Strange non-English characters" above for how to run `doconce change_encoding` to change the encoding of the DocOnce file.

1.37 Too short underlining of reST headlines

This may happen if there is a paragraph heading without proceeding text before some section heading.

1.38 Found `!bt` but no tex blocks extracted (BUG)

This message points to a bug, but has been resolved by removing blank lines between the text and the first `!bt` (inserting the blanks again did not trigger the error message again...).

1.39 Examples are typset with environment delimiters visible

If you see an Example section containing `!bsubex`, `!bsol`, or other begin and end tags for environments, it means that you have intended to typeset examples as exercises, but forgotten the command-line option `--examples_as_exercises`. The text in the example is typeset as is unless this option is included.

1.40 Emacs editing does not work properly because of "regexp overflow"

Sometimes the Doonce editing mode (see the *Emacs DocOnce Formatter* section in the manual) in Emacs leads to an error message ending with “overflow in regexp matcher”. This error is due to some regular expression used in the DocOnce editing mode. The remedy is to split the file into smaller pieces and include the pieces using the `preprocess` directive `#include "piece.do.txt"`. The error message comes with the DocOnce file contains too much text for Emacs to handle.

1.41 My machine hangs if I have many movies

DocOnce has no limits on the amount of movies. When the output is in HTML, one big HTML file may contain too many movies (local movie files or embedded YouTube movies) for the browser to handle. The remedy is to split the document into smaller pieces by inserting

```
!split
```

for every new page. After `doconce format html mydoc`, run `doconce split_html mydoc.html` to get the document split into a main document `mydoc.html` and pieces `._mydocXXX.html`, where `XXX` stands for three digits (`000`, `001`, `002`, and so forth).

1.42 How can I use quotes in a link?

Links are typeset inside double quotes, but DocOnce applies double backticks and double single quotes to typeset quotes, so the right form is

```
"This is a ‘‘link text’’ to google": "http://google.com".
```

It appears as (typset in a `quote` admon):

> This is a “link text” to google.

1.43 Conversion from DocOnce to Google Docs

- Transform the DocOnce document to HTML, upload to Google Drive, right-click the file and open as Google Docs.
- It might be necessary to adjust formatting, e.g., insert an extra line between paragraphs.
- MathJax code is not converted to the Google Docs LaTeX counterpart and appears verbatim.
- Pygmentized computer code and admons look fine.

1.44 Convesion from Google Docs to DocOnce

- Save as HTML file and use Pandoc to convert to Markdown. Use the `-markdown` and `--md2do_output=` options to 'doconce format to convert to DocOnce.
- Lists in Markdown must be intended.
- Notes in Google Docs become (potentially many) footnotes. Some tailored editing is necessary.

This solution is not much explored and more text transformations from Pandoc-generated Markdown is certainly needed.

1.45 Are there any tools for shared online writing of DocOnce documents?

In theory, `http://draftin.com` and `https://stackedit.io` can be used to share Markdown documents and these can be transform to and from DocOnce documents. Tested to a little degree, but may work for very simple documents (sections, lists, code - no labels, refs, math, admons, code copied from file). Not all of Extended Markdown is interpreted by DocOnce, and DocOnce transforms to Panddoc-extended Markdown, not the Extended Markdown used by these sites.

1.46 Examples on seemingly legal syntax that fails

Since DocOnce applies regular expressions to a large extent to translate the input source to the output format, limitations of regular expressions may lead to unexpected results. Here are some examples on what can go wrong.

Double bold/italic. Two strings with bold or emphasize after each other will not work:

```
# Not properly interpreted:
Here is a _bold_ _word_.
# This is how to do it:
Here is a _bold word_.

# Same with emphasize/italic:
Cannot write *two emphasized* *words*,
but must write *two emphasized words*.
```

The wrong syntax implies that one of the texts inside bold/italic tags will appear with DocOnce syntax in the output.

Mix of bold/italic and math. Trying to typeset a mixture of text and mathematics or code in boldface or emphasized font fails:

```
_It is important that $u_1=2$!_
*It is important that `a*b=b*a` in any computer language*.
```

The reason is that boldface or emphasize text cannot contain the special characters dollar sign and backtick. However, if these were allowed, the above examples would fail because the underscore in `$u_1=2$` would mark the end of the boldface text. Similarly, the `*` in `a*b` would mark the end of the emphasized text. Such problems are avoided by *only using plain text inside emphasize or boldface tags*:

```
_It is important that_ $u_1=2$!
*It is important that* `a*b=b*a` *in any computer language*.
```

Another similar example is the mix of `\textcolor{col}{text}` syntax with braces inside the text. For example,

```
color{blue}{However, here a blue color specification
fails: $\frac{1}{2}\omega^2$.}
```

The text part is interpreted as string with `However` and lasting up to the first right brace, which is in `\frac{1}`. A remedy is, as above, to only use plain text inside the text part of the color specification and use a `\textcolor` command inside the mathematics:

```
color{blue}{However, here a blue color specification
fails:} $\textcolor{red}{\frac{1}{2}\omega^2}$.}
```

Mathematics and code in blocks are invisible when inline tagging is interpreted and translated. Therefore, a specification as follows works well:

```
color{red}{This equation,

!bt
\begin{equation}
a = b
label{eq1}
\end{equation}
!et
is meant to be in red and it works.}
```

The rules are that `*text*`, `_text_`, and `\textcolor{color}{text}` employs a simple rule: `text` lasts up to the first end-tag character (`*`, `_`, and `}` above), but code and math blocks do not count.

1.47 How can I make pressbooks.com books from DocOnce?

http://pressbooks.com supports HTML input with WordPress LaTeX mathematics, so the easiest way is to compile DocOnce to HTML with the `-wordpress` option, put the file on the web somewhere, click *Utilities* - *Import* in pressbooks.com, then choose HTML input, write the URL, and upload.

Note that WordPress LaTeX cannot refer to equations using labels! This is a major problem with pressbooks.com.

1.48 How can I produce ePub books from DocOnce?

DocOnce has no direct translation to ePub, but one can apply a script `ebookmaker.py` to translate a set of split HTML files to ePub. Bootstrap styles with mathematics and code come out very nice in ePub this way. The script needs a JSON file. If `mydoc.do.txt` is the DocOnce file and `._mydoc*.html` the set of split HTML files after translation to HTML, the JSON file may look like this:

```
{
    "filename" : "mydoc",
    "title" : "Title of the document",
    "authors" : [
        {
            "name" : "Hans Petter Langtangen",
            "sort" : "Langtangen, Hans Petter"
        }
    ],
    "rights" : "Public Domain",
    "language" : "en",
    "publisher": "hpl",
    "subjects" : [ "Science" ],
    "contributors" : [
        {
            "name" : "Hans Petter Langtangen",
            "role" : "author"
        }
    ],
    "identifier" : {
        "scheme" : "URL",
        "value" : "http://somewhere.net"
    },
    "contents" : [
        {
            "type" : "text",
            "source" : "._mydoc*.html"
        }
    ],
    "toc" : {
        "depth" : 2,
        "parse" : [ "text" ],
        "generate" : {
            "title" : "Index"
        }
    }
}
```

Just edit this file to your needs, save it as `mydoc.json` and run `epubmaker.py mydoc.json` to produce `mydoc.epub`. Read in Calibre on computers or use ebook readers on phones and tablets.

1.49 What about non-English languages and DocOnce?

DocOnce can handle non-English language, although most of its use has concerned documents in English. When writing in languages with non-ASCII characters, remember to use the command-line option `-encoding=utf-8`. Some other points of importance:

- The abstract requires a heading Summary or Abstract. This heading must be substitued by the appropriate word after translation.
- LaTeX files should make use of packages and options for the language in question such that LaTeX-generated headings get the right language-specific wording. Use `doconce replace` or `doconce search` modify the `.tex` file (see the Bash script referred to below for examples).
- Some DocOnce-generated headings are in English, e.g., "Table of Contents", which must be edited by `doconce replace`.
- For Sphinx output, two edits in files in the Sphinx root directory (`sphinx-rootdir` by default) are necessary (see Bash script below for examples on autoediting):
  - Insert right language in `conf.py`: `language = 'nb_NO'` for Norwegian
  - Edit headings in `index.rst`: "Contents:" and "Tables and indices", to what the language demands
  - Unfortunately, "Page contents", "Next page", and "Previous page" remains in English in the navigation area (these can, of course, be replaced in all the HTML files generated by Sphinx in `_build/html` - recall to edit also all the files starting with a dot if you have split the document (`doconce split_rst`))

There is an example of a document in Norwegian in the DocOnce repository where we do all necessary adjustments mentioned above:

- DocOnce source
- Bash script for compiling the DocOnce source
- PDF
- HTML
- Sphinx

1.50 Lists look wrong

It is very important to be accurate with white space in lists. There must be exactly *one blank* between the `o` or `*` that starts an item and the following text, and multiple lines in an item must be perfectly aligned. Here are some typical errors:

```
o a rectangular pulse
o  a Gaussian pulse
o one period of a cosine pulse
o half a period of
 a cosine pulse
```

Problem: `a Gaussian pulse` is not properly aligned, `a cosine pulse` on the last line is not properly aligned. The correct formatting is

```
o a rectangular pulse
o a Gaussian pulse
o one period of a cosine pulse
o half a period of
  a cosine pulse
```

1.51 How can I get an overview of all files that are included in a document?

Suppose the root (main) document is `mydoc.do.txt`. Run

```
Terminal> doconce include_map mydoc.do.txt
```

The output lists all the recursive `#include` Preprocess statements so you can see how all files are included.

1.52 How to write the dollar sign for an amount in dollars?

If you want to write `$1,000`, this is not trivial, because the dollar sign is used for inline math formulas, and everything from this dollar sign up to the next is interpreted as a math formula.

The solution in DocOnce is to prefix the dollar sign by a backslash (same syntax as required by LaTeX). This ensures correct output in all output formats.

Another solution is to pre- or postfix the amount by USD instead of the dollar sign.

1.53 Problems with Externaldocuments specifications

The `# Externaldocuments:` command is used for *generalized references* to lables in other documents. The command specifies directories with `.aux` files and provides the necessary information for using `xr` package for external references in case of LaTeX output. For all other formts an alternative user-given text is used and `# Externaldocuments:` has no effect. A typical use case is that one book refers to another book. This may cause trouble when the `.tex` and `.aux` files specified in the `# Externaldocuments:` command do not exist. A quick fix is to follow the advice in the error message and use the `touch` command to create empty files so DocOnce does not abort the execution. However, the only real fix is to install all the directory trees specified in the `# Externaldocuments:` command.

References to such external documents may also cause trouble when sending the `.tex` files for a book to a publisher, since the `\externaldocument{}` commands in the `.tex` file in this file will refer to `.aux` files in other directories (which the publisher then must have). A good fix to this problem is provided by the sample packing script `pack_Springer.sh`, which calls up another script

grab_stylefiles.py, which automatically copies the .aux files in question to a subdirectory and autoedits the main .tex for the DocOnce document accordingly. The purpose of pack_Springer.sh is to make a directory tree with all files (including stylefiles) needed to compile a LaTeX version of the document. This tree can then be sent to the publisher and is self contained.

Another fix for problems with the # Externaldocuments: command is to hardcode references when files needs to be sent to a publisher. Typically, one has

```
% if not FOR_PUBLISHER:
# Externaldocuments: ../../some/path, ../some/other/path
% endif
```

Every generalized reference to a label in the specified external documents must have a special case when files are meant for a publisher:

```
% if FOR_PUBLISHER
ref[Section 4.2][ in
% else:
ref[Section ref{my:label}][ in
%endif
cite {book}][the section ‘‘Guidelines’’ in the document
"A note on everything": "http://..."].
```

Setting FOR_PUBLISHER=True as part of the doconce format pdflatex command will then effectively remove everything with the # Externaldocuments: command and generalized references.

1.54 What is the difference between doconce replace and doconce subst?

doconce replace replaces one text by another, exactly as the text is written. For example, doconce replace float double *.do.txt is the same as invoking an editor and replacing float by double in all files *.do.txt. On the other hand, doconce subst works with *regular expressions*, so doconce subst from to is the same as re.sub(from, to) in Python. You need experience with re.sub and regular expressions to take advantage of doconce subst. Regular expressions are much more powerful for editing text, but also more complicated and errorprone.

2 Preprocess/Mako

2.1 List important things to remember when programming Mako

- Always remember a colon after Mako control statements like % if VAR: and % else:.

- Keep Python code more than a few lines *outside* of the DocOnce file, and use `# #include` to include the code. See the *Debugging Python code in Mako* section in the manual for how to do it.
- Do not use continuation character (backslash) in Python code.
- When a Mako error refers to a line in the text, invoke the file that Mako sees: `tmp_preprocess__mydoc.do.txt` if the DocOnce file has name `mydoc.do.txt`.
- Use double `##` (Mako comment) to comment out Mako calls to functions or Mako variables.

2.2 The Mako preprocessor is fooled by DocOnce text

Here are possible problems for Mako:

- Strings with `'T<%.1f'` look as openings of Mako blocks (<%); change to `'T < %.1f'` to avoid this confusion.

2.3 The Mako preprocessor has problems with lines starting with percent sign

If you have lines starting with % inside code segments (for example, SWIG code or Matlab comment lines), the Mako preprocessor will crash because it thinks these lines are Mako statements. DocOnce detects this problem and emits an error message. The solution to this problem is to replace a single % at the *very* beginning of the line (column 1) inside code by a double %%. Mako will replace it by a single %. However, contrary to what is claimed in the Mako documentation, an indented double %% remains %% and is not converted to % (and using a single % gives a Mako error). The trick is to write `${'%'}` as demonstrated here:

```
!bc mcod
%% Comment on the beginning of the line can be escaped by %%
if a > b
  ${'%'} Indented comment needs this trick
  c = a + b
end
!ec
```

2.4 The Mako preprocessor gives syntax error in Python code

The information with respect to syntax errors in Python code is sparse. It is recommended to move the Python code to separate files and test it with the ordinary Python interpreter. See the section *Debugging Python code in Mako* in the DocOnce manual.

2.5 The Mako preprocessor gives strange error messages

If you to a little syntax error in Mako, the consequences can be quite unpredictable. Especially, if you forget curly braces around variables or function calls, the forthcoming text is processed as part of the Mako command. Pay attention to the line number reported by Mako: this is the line number after Preprocess has processed the DocOnce document, so you need to load `tmp_preprocess__mydoc.do.txt` and go to the right line in that file to see the Mako problem (here the DocOnce document is named `mydoc.do.txt`).

Look through the specific Mako problems reported below, and if they do not bring you to a solution of the problem, search for all occurences of dollar, left curly brace and check the syntax carefully. Likewise, check the syntax of Mako if-tests.

Another widely used technique is to copy out small parts of the complete document to a separate file and run `doconce format`. In this way you can easier see which parts of the document that work and where the error suddenly appears.

Tip: Compile your DocOnce document frequently.

The best means against Mako problems is to run `doconce format` often. Combined Git, you can take a `git diff` to see what you have recently changed and get an idea what can be wrong.

2.6 The Mako preprocessor claims a variable is undefined

Very often such errors are related to typos when using Mako variables or functions, or correct yet undesired LaTeX syntax. For example,

```
${\cal O}(\Delta x^2)$
```

is valid LaTeX, but the dollar sign and curly braces confuse Mako. Rewrite such mathematics. It is wise to not use `${` anywhere in LaTeX mathematics. Create a newcommand if there are no other possible rewritings. A known common problem is `${}^+$` type of indication of superscripts. A suggested rewrite is `$\,{}^+$`.

The error message will ask you to rerun `doconce` with `--mako_strict_undefined`, which will display the variable that is confusing Mako. Sometimes the variable is printed, sometimes a totally different name is said to be undefined. This is confusing, because then you have to use the bisection method below to narrow down the problem yourself.

Do not continue to use `--mako_strict_undefined` while you are debugging because this variable or a new variable will then always be undefined in that mode. Rerun without `--mako_strict_undefined` to see if the problem is gone.

If not, try the option again, and if no progress, use `# #ifdef` directives to comment out large portions of the text and apply a bisection procedure to locate where the Mako problem is (without `--mako_strict_undefined`). A bisection procedure means that you comment out the last half, find in which half the problem is, comment out half of that half, find in which half the problem is, and so on. The procedure converges pretty quickly, even for large books.

2.7 Something goes wrong in the preprocessing step

You can examine `tmp_preprocess__filename` and `tmp_mako__filename`, where `filename` is the complete name of the DocOnce file, to see what the preprocessors actually to and if something is wrong in these files before DocOnce starts translating the text. One or both of those files may be missing, but examine the beginning of the output from DocOnce to see exactly which preprocessors are run and on which files.

2.8 Preprocessor directives do not work

Make sure the preprocessor instructions, in Preprocess or Mako, have correct syntax. Also make sure that you do not mix Preprocess and Mako instructions. DocOnce will then only run Preprocess.

3 Code or Tex Blocks

3.1 Too long lines in math blocks

A common problem in LaTeX (not directly related to DocOnce) is too long lines in math blocks. The trick is to use the `align` or `align*` environment and introduce newlines with `\nonumber\\` to break up lines (without introducing an equation number). Typically,

1. Insert `\nonumber\\` where you see a linebreak should occur.
2. Start next line with `&\quad` such that the continuing line starts slightly to the right of the alignment character used to align the equations. If you have only one equation, find some point where you insert `&` and just start the next line with `&` - no `\quad` displacement to the right is needed in that case.
3. `\left(` and `\right)` (and similar constructions for automatic adjustment of the sizes of parentheses) do not work accross linebreaks. This means that you must replace `\left(` by (e.g.) `\biggl(` unless there is a corresponding `\right` on the same line. Whether to choose `\biggl`, `\bigl`, or `\Biggl` depends on the desired size of the the parenthesis.

3.2 Code or math block errors in reST

First note that a code or math block must come after some plain sentence (at least for successful output in reST), not directly after a section/paragraph heading, table, comment, figure, or movie, because the code or math block is indented and then become parts of such constructions. Either the block becomes invisible or error messages are issued.

Sometimes reST reports an "Unexpected indentation" at the beginning of a code block. If you see a `!bc`, which should have been removed when running `doconce format sphinx`, it is usually an error in the DocOnce source, or a problem with the rst/sphinx translator. Check if the line before the code block ends in one colon (not two!), a question mark, an exclamation mark, a comma, a period, or just a newline/space after text. If not, make sure that the ending is among the mentioned. Then `!bc` will most likely be replaced and a double colon at the preceding line will appear (which is the right way in reST to indicate a verbatim block of text).

3.3 Strange errors around code or TeX blocks in reST

If `idx` commands for defining indices are placed inside paragraphs, and especially right before a code block, the reST translator (`rst` and `sphinx` formats) may get confused and produce strange code blocks that cause errors when the reST text is transformed to other formats. The remedy is to define items for the index outside paragraphs.

3.4 Something is wrong with a verbatim code block

Check first that there is a "normal" sentence right before the block (this is important for reST and similar "ASCII-close" formats).

3.5 Code/TeX block is not shown in reST format

A comment right before a code or tex block will treat the whole block also as a comment. It is important that there is normal running text right before `!bt` and `!bc` environments.

3.6 Verbatim code blocks, figures and admons inside lists look ugly

Avoid verbatim code blocks, figures, and admons inside lists (it makes life easier!). Instead, use paragraph headings, say

Step 1. Describe step. Add figure, code block, admon, etc.

Step 2. Describe next step.

If you miss the automatic numbering of items in enumerated lists, you can simulate that with a Mako variable:

```
<% counter = 0 %>

<% counter += 1 %>
__Step ${counter}.__
Describe step. Add figure, code block, admon, etc.

<% counter += 1 %>
__Step ${counter}.__
Describe next step.
```

3.7 LaTeX code blocks inside lists look ugly

Same solution as for computer code blocks as described in the previous paragraph. Make sure the `!bt` and `!et` tags are in column 1 and that the rest of the non-LaTeX surrounding text is correctly indented. Using paragraphs instead of list items is a good idea also here.

3.8 LaTeX code appears prior to paragraph headings

If you have a paragraph heading and then code right after,

```
__Heading.__

!bc pycod
def f(x):
    return 42
!ec
```

the generated LaTeX code is correct, but it is translated to PDF in a wrong way such that the heading appears *after* the code. A remedy is to insert some text after the paragraph heading. See also Section 5.29 - it is the same problem.

3.9 Numbered equations in exercise solutions change the equation numbers in the text...

This is a problem if you produce versions with and without solutions, see the heading “Numbering of Extra Equations in Solutions” in the DocOnce manual for description of the problem and its solutions.

4 reST/Sphinx

4.1 Math formulas are not rendered properly

HTML generated by Sphinx applies MathJax for rendering LaTeX mathematics. Due to the way Sphinx embeds these formulas in HTML, sometimes the formu-

las are not being rendered and instead the pure LaTeX code is put in a box. The same formulas may appear right in pure HTML.

The first thing to do is to check that the LaTeX generated PDF format shows the formula correctly. The next step is to compile to HTML and check if MathJax in pure HTML renders the formula correctly. If so, the formula is correctly typeset, and the problem is related to Sphinx.

There are basically two remedies: either drop Sphinx and go for HTML, or try to rewrite the formula and try until the Sphinx-generated HTML code manages to render it.

Copy the problematic typesetting to a minimalistic document. If the original document is split into pieces via `doconce split_rst`, make sure your minimalistic document also contains a `!split` command and is split into at least two pieces. This is because equations get tags when documents are split, and with the `\tag{}` command, it seems that sphinx rewrites the MathJax code, and this may cause problems. This also means that dropping `doconce split_rst` and having one big HTML file for Sphinx can make the rendering of mathematics (much) easier.

Some typical problems encountered are listed below.

- Formulas with use of `\bar\boldsymbol{}`.
- Align environments with multiple equations. Try to simplify typesetting.
- `\mbox` or `\hbox` commands in equations.

4.2 Sphinx directory generation aborts with "label empty or too long"

This is a strange error that might arise from a too long title of the document. The solution is to specify a shorter title when running `doconce sphinx_dir` (i.e., to avoid getting the title from the original `.do.txt` document):

```
Terminal> doconce sphinx_dir title="Short Title" \
          authors="A. B. Crunch" theme=pyramid mydoc
```

4.3 Title level inconsistent

reST does not like jumps in the levels of headings. For example, you cannot have a `===` (paragraph) heading after a `=======` (section) heading without a `====` (subsection) heading in between.

4.4 Subsection headings appear at the level of sections in the table of contents

This is a problem that arises if you split a Sphinx document at the subsection level. Then the subsection is the highest level in that part, and it will then move

up to the section level in the table of contents (this is done by Sphinx, and DocOnce has no influence on the process).

After Oct 2014, DocOnce ignores by default the `!split` commands that the user has inserted and performs a split at the highest section level, which means that chapters constitute the parts of the document, if chapters are present, otherwise the sections represent the different parts. If this behavior is not desired and the user's `!split` commands are to be respected, provide the command-line argument `--sphinx_keep_splits` in the `doconce format` command.

4.5 Lists do not appear in .rst files

Check if you have a comment right above the list. That comment will include the list if the list is indentend. Remove the comment.

4.6 Error message "Undefined substitution..." from reST

This may happen if there is much inline math in the text. reST cannot understand inline LaTeX commands and interprets them as illegal code. Just ignore these error messages.

4.7 Warning about duplicate link names

Link names should be unique, but if (e.g.) "file" is used as link text several places in a reST file, the links still work. The warning can therefore be ignorned.

4.8 Inconsistent headings in reST

The `rst2*.py` and Sphinx converters abort if the headers of sections are not consistent, i.e., a subsection must come under a section, and a subsubsection must come under a subsection (you cannot have a subsubsection directly under a section). Search for `===`, count the number of equality signs (or underscores if you use that) and make sure they decrease by two every time a lower level is encountered.

4.9 No code environment appears before "bc ipy" blocks

The `!bc ipy` directive behaves this way for `sphinx` output because interactive sessions are automatically handled. If this is not appropriate, shift to `!bc cod` or another specification of the verbatim environment.

5 LaTeX

5.1 Lines are too long

Too long lines are often handled with a forced newline like \\ in LaTeX, but this double backslash does not look good in other formats when you use DocOnce. Try to rewrite the text such that LaTeX naturally avoids too long lines. Inline verbatim can easily force LaTeX to extend the line width, but see if you can use other words to move the inline verbatim to the left or to the next line.

A last resort is to use the preprocessor and test on "FORMAT == 'pdflatex'" and then use the double backslash.

5.2 Tables get too wide - need to wrap text in the columns

One can use the `tabularx` package. This is automatically enabled by specifying the alignment of the actual column as `X` (means `l` for all other output formats), for instance `|--c--X--|`.

5.3 Table inside user-defined environment does not work

If the table appears at the end of the environment (e.g., a theorem or example environment), add a sentence between the table and the end-of-environment command.

5.4 Strange `begindescription` error arising from math

If you have an inline math formula that is split over two lines, and the formula continues with a minus sign on a line, this minus sign may be interpreted as a *description list* and typeset as a list, which is nonsense. The remedy is to move the minus sign at the beginning of the line to the end of the last line.

5.5 Inline figures should have more vertical space

You can insert `<linebreak>` (which gives a vertical space if alone on a line):

```
<linebreak>
<linebreak>

FIGURE: [...]

<linebreak>
<linebreak>
```

5.6 How can I use a special package for typesetting code?K (not `pyg`, `lst`, or `vrb`)

One gets the impression that `--latex_code_style=` has only the options for Pygments, Listings, or the Verbatim packages, but in fact any package can be used.

Suppose you have the fictitious code enviroment `ultimate` from the package `compcode` that you want to use, together with a yellow backgrund, and with some environment-specific parameters `arg1` and `arg2`. Then you can write

```
Terminal> doconce format pdflatex mydoc --latex_packages=compcode \
          '--latex_code_style=ultimate-yellow2[arg1=val1,arg2=val2]'
```

The resulting LaTeX code in `mydoc.tex` becomes

```
\begin{cod}{cbg_yellow2}\begin{ultimate}[arg1=val1,arg2=val2]
...
\end{ultimate}\end{cod}
```

In this way, you can use any environment in any package for typesetting code.

5.7 Sphinx and HTML show pygmetized code, but not LaTeX

You have to explicitly select the `minted` environment when running `ptex2tex` to get pygmentized computer code in LaTeX:

```
Terminal> doconce ptex2tex mydoc envir=minted
Terminal> pdflatex -shell-escape mydoc
```

5.8 LaTeX error because of underscores

LaTeX requires prefixing underscores by a backslash. Either typeset words with underscores in inline verbatim text (so you do not need the backslash) or use the preprocessor to insert LaTeX-specific text with a backslash, e.g.,

```
Here is a word underscore:
%% if FORMAT in ('latex', 'pdflatex'):
my\_word
%% else
my_word
%% endif
```

Usually, inline verbatim text is way to go with underscores anyway.

5.9 LaTeX does not like underscores in URLs

Suppose you have a URL reference like

```
..which can be found in the file "my_file.txt":
"http://some.where.net/web/dir/my_file.txt".
```

LaTeX will stop with a message about a missing dollar sign. The reason is that underscores in link texts need to be preceded by a backslash. However, this is incovenient to do in the DocOnce source since the underscore is misleading in other formats. The remedy is to format the link text with inline verbatim tags (backticks):

```
..which can be found in the file "‘my_file.txt‘":
"http://some.where.net/web/dir/my_file.txt".
```

Verbatim text in links works fine with underscores.

5.10 Inline verbatim fails in admon headings

A known problem is to have the word

```
\Verb
```

in (admon) headings. Otherwise inline verbatim should work, both in section headings and admon headings.

5.11 How can I customize the look of an admonition?

The default environment, `--latex_admon=mdfbox`, has many options for customization. Say we want the notice environment to have a very small font, no background color, no line below the title, no visible frame, and a smaller width. All this is a matter of editing the variables in the `newmdenv` command that defines the `notice_mdfboxmdframed` environment in the `.tex` file.

The simplest approach is to make a script that automatically edits some of the variables in `newmdenv` environments:

```
name=tmp1
# Edits for mdframed environments
doconce format pdflatex $name --latex_code_style=vrb
doconce subst '  backgroundcolor=.+' \
        '  backgroundcolor=white,' $name.tex
doconce subst '  frametitlebackgroundcolor=.+' \
        '  frametitlebackgroundcolor=white,' $name.tex
doconce subst '  leftmargin=.+' '  leftmargin=3cm,' $name.tex
doconce subst '  rightmargin=.+' \
        '  rightmargin=3cm,\n  font=\\tiny,' $name.tex
doconce subst '  frametitlerule=.+' \
        '  frametitlerule=false,' $name.tex
doconce subst '  linewidth=.+' '  linewidth=0pt,' $name.tex
# Edits for fontsize in Verbatim
doconce replace ',fontsize=\fontsize{9pt}{9pt}' \
        ',fontsize=\fontsize{6pt}{6pt}' $name.tex
```

Note that these edits actually apply to all types of admonitions! More tailored edits must include the text of the entire environment definition.

Here is a possible test file:

```
======= Heading =======

Bla-bla.


\begin{notice_mdfboxadmon}[Long box.]
% for i in range(10):
bla-bla, bla-bla, bla-bla, bla-bla, bla-bla, bla-bla, bla-bla,
% endfor

!bt
\begin{align}
x &\in [0,1]\\
y &= sin(x)
\end{align}
!et

!bc pycod
def myfuncf(x):
    return sin(x)
!ec

% for i in range(100):
bla-bla, bla-bla, bla-bla, bla-bla, bla-bla, bla-bla, bla-bla,
% endfor
\end{notice_mdfboxadmon} % title: Long box.
```

A (much) better approach is to define a *user-defined environment*, called (e.g.) minipage, see the manual. This environment has begin tag `!bu-minipage` and end tag `!eu-minipage`. The environment is defined in a file `userdef_environments.py` in the same directory as the DocOnce source file. An example is given in this `userdef_environments.py` file.

5.12 How can I have unnumbered sections?

Use the `--section_numbering=off` option:

```
Terminal> doconce format pdflatex mydoc --section_numbering=off
```

5.13 LaTeX suddenly places figures differently

Until August 25, 2015, the figure environment applied the `[t]` (top) option, but since then the `[h]` option is used. Of even greater influence are the `topfraction`, `bottomfraction`, and `textfraction` parameters in the `.tex` file. See the "Figures" section in the manual for more information and how to customize the placement of figures.

5.14 I want figure references with page number too

A DocOnce user wanted figure references to be as follows in LaTeX:

```
% if current page is different from figure page:
... Figure \ref{my:fig} on page \pageref{my:fig}.
% if current page is the same as the figure page:
... Figure \ref{my:fig}.
```

All other formats should emit the standard treatment of a DocOnce figure reference with `ref`.

Quick solution with the `varioref` package. DocOnce actually generates such references in LaTeX output if you request the `varioref` package: `--latex_packages=varioref` (the standard `ref` command is now replaced by `\vref`, except in equation references).

Manual solution (or a good example on what Mako can do). Here is an implementation using if-else tests in LaTeX and internal variables. Unfortunately, this solution is not reliable because it uses the `\thepage` variable in LaTeX (for the page number of the current page) and this variable is unreliable. The solution above (using `\vref` and the `varioref` package) is much better. However, the manual solution is a good example on how to use Mako to do seemingly complicated things.

The above problem can be solved by some Mako programming and usage of the LaTeX packages `ifthen` and `refcount`.

```
<%
def figref(label):
    text = 'Figure ref{%s}' % label
    if FORMAT in ('latex', 'pdflatex'):
        text += r'\ifthenelse{\equal{\thepage}{\getpagerefnumber{%s}}}{}{ on page \pageref{%s}}' % (l
    return text
%>

FIGURE: [myfigfile, width=600 frac=0.8] My figure. label{my:fig}

We refer to ${figref('my:fig')}.

...
# Much later
..., as seen in ${figref('my:fig')}.
```

The first reference is on the same page as the figure, and LaTeX should not write the page number, while the second reference is on another page so both the figure number and the page number should appear in the output. All other formats have a standard figure reference (translated to a figure number in HTML and to the caption in Sphinx).

The LaTeX output is something like this:

```
\begin{figure}[h] % my:fig1
  \centerline{\includegraphics[width=0.8\linewidth]{myfigfile.png}}
  \caption{
  My figure. \label{my:fig1}
  }
\end{figure}
```

```
%\clearpage % flush figures sec:fig

We refer to Figure~\ref{my:fig1}\ifthenelse{
\equal{\thepage}{\getpagerefnumber{my:fig}}}{}{ on page \pageref{my:fig1}}.

...
% Much later
..., as seen in Figure~\ref{my:fig1}\ifthenelse{
\equal{\thepage}{\getpagerefnumber{my:fig}}}{}{ on page \pageref{my:fig1}}.
```

5.15 Error when running latex: You must have 'pygmentize' installed

This message points to the use of the minted style for typesetting verbatim code. You need to include the `-shell-escape` command-line argument when running `latex` or `pdflatex`:

```
Terminal> latex -shell-escape file mydoc.tex
Terminal> pdflatex -shell-escape file mydoc.tex
```

Using `doconce ptex2tex` will turn on the minted style if specified as environment on the command line, while using `ptex2tex` requires the preprocessor option `-DMINTED` to turn on the minted package. When this package is included, `latex` or `pdflatex` runs the `pygmentize` program and the `shell-escape` option is required.

5.16 Why are the LaTeX section headings smaller than normal?

DocOnce inserts a special command to make the headings more compact:

```
\usepackage[compact]{titlesec}
```

as explained in the `titlesec` package documentation. To retrieve the standard LaTeX headings, comment out this line or remove it:

```
Terminal> doconce format pdflatex mydoc
Terminal> doconce subst '\\usepack.+\{titlesec\} '' mydoc.p.tex
```

You can easily make the headings even smaller than the normal font by replacing `[compact]` by `[compact,small]` as parameter specification for `titlesec`.

5.17 I get LaTeX compilation errors about "shadedquoteBlue" in code blocks in exercises

When using colored boxes for code in LaTeX, a code snippet is needed in the ordinary running text *before* code snippets inside exercises, problems, or projects. This is a kind of bug in DocOnce that is challenging to fix. It is usually only a problem when writing documents mainly containing exercises, problems, or projects.

5.18 I get compilation error due to pause commands

The `|\pause|` commands are intended for popping up code segments in slides. If you compile the document without having run `doconce slides_beamer` first, standard LaTeX have a problem with `|\pause|`. The remedy is to just remove these commands by

```
Terminal> doconce subst '\|\\pause\|\n' '' mydoc.tex
```

before running `pdflatex mydoc`.

5.19 The cell structure in slides does not work for me ...

Common errors:

- Empty cells? No cell can be empty. Just insert a comment `# Empty cell` to make it non-empty.
- Wrong numbering of the cells. They go 00, 01, 02, 10, 11, etc.

Also, text outside the slidecell boxes *are not shown* in the slide. Everything you want to see must be inside the cells. Use `<linebreak>` repeatedly to insert vertical spacings inside cells.

5.20 Can I have LaTeX figures with shadows?

This is easy by including the `fancybox` and `graphicx` packages and wrapping all `\includegraphics` in a shadow box:

```
Terminal> doconce format pdflatex mydoc
Terminal> doconce replace \
          'microtype}', 'microtype,fancybox,graphicx}' mydoc.p.tex
Terminal> doconce subst '(\\includegraphics\[.+\}\})' \
          '\\shadowbox{\g<1>}' mydoc.p.tex
```

5.21 How can I use my fancy LaTeX environments?

See the section *Example: Defining a Theorem Environment* in the DocOnce manual for how to make a custom theorem environment also in DocOnce, without using implementations restricted to LaTeX. See also Section 1.14.

5.22 The LaTeX file does not compile

If the problem is undefined control sequence involving

```
\texttt{...}
```

the cause is usually a verbatim inline text (in back-ticks in the DocOnce file) spans more than one line. Make sure, in the DocOnce source, that all inline verbatim text appears on the same line.

5.23 The LaTeX Beamer file does not compile

Make sure you have a `!split` before every slide heading.

5.24 Inline verbatim gives error

Check if the inline verbatim contains typical LaTeX commands, e.g.,

```
some text with ‘\usepackage{mypack}‘ is difficult because
ptex2tex will replace this by \Verb!\usepackage{mypack}! and
then replace this by
{\fontsize{10pt}{10pt}\verb!\usepackage{mypack!}}
which is wrong because ptex2tex applies regex that don’t
capture the second }
```

The remedy is to place verbatim LaTeX commands in verbatim blocks - that is safe.

5.25 Errors in figure captions

Such errors typically arise from unbalanced curly braces, or dollar signs around math, and similar LaTeX syntax errors.

(Note that verbatim font is likely to cause trouble inside figure captions, but DocOnce will automatically replace verbatim text in back-ticks by a proper `texttt` command (since verbatim font constructions does not work inside figure captions) and precede underscores by backslash.)

5.26 I want to tune the top (preamble) of the LaTeX file

It is easy to provide a customized preamble text:

```
Terminal> doconce format latex mydoc --latex_preamble=mytop.tex
```

If `mytop.tex` starts with a `documentclass` specification, the whole file becomes the complete preamble in `mydoc.p.tex`, otherwise `mytop.tex` is added at the end of the preamble generated by DocOnce.

If you already have some preamble that defines styles etc., I recommend to generate a `.tex` file from DocOnce and merge the preamble you have with the one generated by DocOnce. When the `.tex` file compiles with the desired preamble, store it in a separate file and use it with the `--latex_preamble=` option. (This approach will for fix the preamble for the future so updates of the DocOnce-generated preamble will not be available for this document.)

Note also that `doconce replace` and `doconce subst` can be used after `doconce format` to tune the LaTeX code in the preamble (and elsewhere). Such automatic edits are useful if they are few.

There are comment lines with

```
--- begin preamble ---d
```

and

```
--- end preamble ---
```

in the generated `.tex` file that can be used to replace the whole preamble by another text via a script.

5.27 How can I control the placement of figures?

In general, LaTeX will control the placement, and DocOnce applies (for now) the `[t]` option to recommend figures to be placed at the top of a page. This can be changed in different ways, using techniques from Sections 1.22 and 5.26. The `float` package and the `[H]` ("here") option for figures makes figures appear closer to their original location:

```
\usepackage{float}
\floatplacement{figure}{H}
```

The `[t]` option in figures must be removed:

```
doconce replace '{figure}[t]' '{figure}' mydoc.tex
```

Custom preamble file. Place the lines above in a file `mypreamble.tex` and ask DocOnce to add this file to the preamble: `--latex_preamble=mypreamble.tex`.

Edit the .tex file. There is a special line in the `.tex` file,

```
%% insert custom LaTeX commands...
```

which can be used to insert desired commands, in this case by

```
doconce subst '% insert custom LaTeX.+' '\\usepackage{float}\n\floatplacement{figure}{H}' mydoc.tex
```

Note that you in both cases need to run the `doconce replace` above command to remove the `[t]` option.

5.28 How can I get a two-column text?

Simply add `twocolum` to the `\documentclass` command. If the `mydoc.tex` file looks like

```
\documentclass[%
twoside,
...
```

you can do a simple substitution edit

```
Terminal> doconce subst 'twoside,' 'twocolumn,\ntwoside,' mydoc.tex
```

5.29 Solution heading in exercise solutions appear after the solution content

Sometimes a solution environment with code

```
!bsol
!bc cod
def f(x):
    return x + 2
!ec
!esol
```

leads to strange behavior with verbatim LaTeX environment: the **Solution** heading appears after the code in the solution. This appears to be a LaTeX problem since the generated LaTeX code has the heading and solution content in the right order. A working remedy is to insert a text before the code:

```
Code:
!bsol
!bc cod
def f(x):
    return x + 2
!ec
!esol
```

5.30 Subexercises (`subexa`) triggers "Missing number, treated as zero"

Check if you have done a `doconce subst` of `section` to `section*`. Such a substitution will alter the newcommand `\subex` defined in the `.tex` file and insert an erroneous `\@startsection*`, which should not contain the `*`. Be more specific with the substitution of `section` in the `.tex` file:

```
doconce subst '\\section\{' '\\section*{' mydoc.tex
doconce subst '\\subsection\{' '\\subsection*{' mydoc.tex
# or
doconce replace '\section{' '\section*{' mydoc.tex
doconce replace '\subsection{' '\subsection*{' mydoc.tex
```

5.31 Page numbers appear in both header and footer

If you experience page numbers in header and footer (this is a problem if you have `{copyright...}` notifications as part of an `AUTHOR` specification), you need to edit the `.tex` file. Removing the page number in the footer is done by

```
Terminal> doconce subst '\\fancyfoot\[LE.+' '' mydoc.tex
```

Page numbers in headers typically arise from using a special LaTeX style or the `doconce format` option `--latex_fancy_header` (but in the latter case there will never be page numbers in the footer).

5.32 Links in the table of contents go to wrong page

This is not an uncommon problem. We resolved this problem for the index by adding `\cleardoublepage\phantomsection` right before `\printindex`. The same trick can be useful in other contexts.

5.33 Exercises in the `Springer_sv` style looks strange

The `Springer_sv` style demands exercises to appear within the `prob` environment, and the `svmono.cls` package typesets `prob` with small headlines.

5.34 How to typeset papers with specified style files?

DocOnce has the `--latex_style=` option for specifying a few LaTeX styles for papers and books. This option mostly support book styles, since traditionally, paper styles in various journals often have special commands for authors, institutions, summary, and so on that are cumbersome to support through DocOnce. It was often easier to develop the paper in DocOnce and at submission time, manually edit the LaTeX output to fit a particular journal style. (SIAM and Elsevier journal have had direct support in DocOnce, though.)

With open access publishing it makes more sense to maintain a paper in DocOnce and publish derived and future versions in various electronic formats. How to then, at some stage in the process, make a special LaTeX version according to a journal's style?

The simplest approach is probably to develop a compilation script that performs a set of edits through `doconce replace` and `doconce subst` commands that turn standard DocOnce LaTeX output into the text required for a special journal style. Usually, such edits are about adding packages, maybe some special commands, and replacing perhaps title, author, date, and summary commands. Below is a sketch how that can be done for PLOS journals. These journals offer a LaTeX template demonstrating appropriate commands. Some text is needed in the preamble, and we can insert that text at the place marked for user-specific extensions ("insert custom LaTeX commands...").

```
#!/bin/bash
name=mydoc

# Compile mydoc.do.txt to latex output mydoc.tex
doconce format pdflatex $name --latex_bibstyle=plos2015 --latex_code_style=vrb

# Edit mydoc.tex file
doconce subst '% insert custom LaTeX commands...' '
\usepackage[aboveskip=1pt,labelfont=bf,labelsep=period,justification=raggedright,singlelinecheck=off]
% Text layout
\raggedright
\setlength{\parindent}{0.5cm}
\textwidth 5.25in
\textheight 8.75in
% Remove brackets from numbering in List of References
\makeatletter
```

```
\renewcommand{\@biblabel}[1]{\quad#1.}
\makeatother
\usepackage[right]{lineno}
\usepackage[top=0.85in,left=2.75in,footskip=0.75in]{geometry}' $name.tex

# Compile the usual way
pdflatex $name
```

This example hopefully gives some ideas how to create specialized LaTeX output from DocOnce documents. If the journal style employs special LaTeX environments, see Section 1.14 how to utilize these from DocOnce.

5.35 I want exercises to have small headings (not subsections)

By default, DocOnce typesets exercises as subsections in LaTeX. However, the heading is defined by a newcommand `exercisesection` (right after `\begin{document}`) so it is easy to switch to any tailored heading. For example, we can use a paragraph heading:

```
Terminal> doconce subst \
          '\{exercisesection\}\[1\]\{\\subsection\*' \
          '{exercisesection}[1]{\paragraph' mydoc.tex
```

This typesetting is perhaps more suitable if you have exercises scattered around in the document rather than grouped in sections.

6 gwiki

6.1 Strange nested lists in gwiki

DocOnce cannot handle nested lists correctly in the gwiki format. Use nonnested lists or edit the `.gwiki` file directly.

6.2 Lists in gwiki look ugly in the gwiki source

Because the Google Code wiki format requires all text of a list item to be on one line, DocOnce simply concatenates lines in that format, and because of the indentation in the original DocOnce text, the gwiki output looks somewhat ugly. The good thing is that this gwiki source is seldom to be looked at - it is the DocOnce source that one edits further.

7 HTML

7.1 How can I add a search facility to an HTML document?

There are many services for this. FreeFind is an easy-to-use service that gives you some HTML code to include in the document, typically inside a `# #if FORMAT == "html"` and `# #endif` block.

7.2 MathJax formulas are not properly rendered

Here are some common problems:

- Two equations cannot have identical label (this error often arises from copying and pasting equations).
- `[` and `]` brackets must sometimes be replaced by `\lbrack` and `\rbrack`.
- Inline mathematics cannot have a space after the first dollar sign.
- Equation environments must be `\[ ...\]`, `equation`, `equation*`, `align`, or `align*` - avoid all other types of environments (`split`, `alignat`, and whatever).
- Errors in formulas that LaTeX accepts may still lead to unrendered formulas in MathJax.

If the corresponding LaTeX-based PDF output has the correct formula, one must try to rewrite the LaTeX mathematics so MathJax accepts it. MathJax can only handle a subset of what is legal mathematics in LaTeX.

Even if HTML is your target output format, it is strongly recommended to develop the mathematics with `pdflatex` as output to debug formulas before you try to render them in MathJax.

7.3 How can I change the layout of the HTML page?

Autoedits of the `.html` file. The easiest way is to edit the HTML style or the HTML code directly. However, those edits are overwritten the next time you compile the DocOnce document to HTML. The edits should therefore be automated using the `doconce subst` (for regular expression editing) or `doconce replace` (for plain text substitution editing) commands in the file where you run `doconce format html mydoc`. For example, say you want narrower gray admonition boxes with a much thicker boundary. The `.alert` style must then be changed, more precisely the `border` and the `width` specification. You can then add these statements after running `doconce format html ...`:

```
doconce replace 'border:1px' 'border:11px' mydoc.html
doconce replace 'width: 75%;' 'width: 35%' mydoc.html
```

Specify a `.css` file. Another way to control the layout is to copy the style in the HTML file into a `.css` file, edit that file as you like, and provide the file as part of the compilation using the `-css=mystyle.css` flag. For example, say you want to transform the headings to something attractive for young girls (white text on pink background). A `.css` file can be written from scratch, but one can also generate one from one of the built-in DocOnce HTML styles by running

```
Terminal> doconce format html mydoc --css=pink.css \
          --html_style=blueish
```

The result is a new file `pink.css` containing the specification of the `blueish` style. One can now change this `pink.css` file to `h1` headings with pink background and white text, `h2` and `h3` headings with pink text, and pink color for visited or hovered hyperlinks:

```
h1 {
    font-family:"comic sans ms";
    width: 900px;
    font-size: 48px;
    text-align: center;
    background: #EA00FF; /* pink */
    border-style: solid;
    border-width: 5px;
    border-color: #FFFFFF;
    padding: 20px;
    margin: 20px;
    margin-left: 10px;
    color: white
}
h2, h3 {
    font-family:"comic sans ms";
    font-size: 18px;
    color: #EA00FF;
}
h3 { font-size: 24px; }
a { color: #1e36ce; text-decoration:none; }
a:hover, a:visited { color: #EA00FF; /* pink */ }
...
```

The rest of the file can remain unaltered. Running

```
Terminal> doconce format html mydoc --css=pink.css \
          --html_style=blueish
```

will now render `mydoc.html` with pink headings and visited links.

Specify a template. The standard of way of completely controlling the HTML format is to use an HTML template. The DocOnce source is then the body of text (leave out `TITLE:` to get HTML without a header and footer). The `--html_template=filename` command-line option will then embed the DocOnce text in the specified template file, where you can use style sheets and desired constructs in the header and footer. The template can have "slots" for a title (`%(title)s`), a date (`%(date)s`), and the main body of text (`%(main)s`).

For typesetting code, `pygments` is used (if installed) and can be turned off by `--no_pygments_html` (leaving code in gray boxes).

The easiest way is to get fancy layouts in HTML is to use the `sphinx` format and one its many themes.

7.4 Why do figures look ugly when using HTML templates?

The HTML header that DocOnce generates contain special styles for figure captions and the horizontal rule above figures. When using templates these styles are not defined, resulting in a rule that spans the width and a centered caption. Changing the appearance of the rule and caption can either be done by inserting styles or simply by automatic editing of the HTML code in a little shell script:

```
doconce replace '<p class="caption">' \
 '<p style="width: 50%; font-style: italic; color: blue">' mydoc.html
doconce replace '<hr class="figure">' \
 '<hr style="width:60%, font-style: normal">' mydoc.html
```

8 reveal/deck HTML5 slides

8.1 Python Online Tutor code does not work in reveal

Use the Chrome browser (the code does not work in Firefox in combination with reveal HTML5, but it works in plain HTML).

8.2 Reveal slides are stacked on top of each other

This problem can be caused by having a `!bblock` environment crossing two slides because of a forgotten `!eblock` on the first slide. To locate the problematic slides, copy chunks of slides to a new file, compile, and inspect. This will reveal the problematic chunk.

8.3 Reveal slides are moving steadily to the left

This seems to be a problem when one has used the mouse to scroll down on a slide and continue to use the right arrow for moving to the next slide. Click on the arrow in the slide instead of using the arrow key.

8.4 YouTube movies do not work

They do not work in reveal or deck unless the Chrome browser is used. Usually the HTML5 slides work best in Firefox.

8.5 Online Python Tutor does not work

- Check that there are no syntax errors in the code by running the code as an ordinary file. Syntax errors lead to empty space and no tutor in HTML output.
- Seems to be a problem with reveal slides. The *Forward* button is not clickable.