DocOnce Troubleshooting

Hans Petter Langtangen[1,2]

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

Sep 2, 2014

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 1.22 gives some examples of what can go wrong.

1 General Problems

1.1 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 for scientific subjects, 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 for scientific subjects, very much hard (and exciting!) work, and *thinking constantly about it*.

Without <linebreak> the rendering becomes

So how to be excellent?.

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

1.2 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`.

1.3 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.

1.4 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.

1.5 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.

1.6 Problems with boldface and emphasize

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

1.7 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.

1.8 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.

1.9 Inline verbatim code is not detected

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

1.10 Inline verbatim text is not formatted correctly

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

1.11 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 UTF-8, convert to latin1 using either of the Unix commands

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

1.12 Wrong Norwegian charcters

When DocOnce documents have characters not in the standard ASCII set, the format of the file must be LATIN1 and not 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.

1.13 Too short underlining of reST headlines

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

1.14 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...).

1.15 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.

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

Sometimes the Doonce editing mode (see Section **??**) 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.

1.17 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).

1.18 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

> This is a “link text” to google.

1.19 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.

1.20 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.

1.21 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.

1.22 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.

2 Preprocess/Mako problems

2.1 List important things to remember when programming Mako

- 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 of the form `${name...}`.

2.2 The Mako preprocessor is seemingly not run

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 avoids running Mako. Examine the output from DocOnce: warnings are issued if Mako is not run.

2.3 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.

2.4 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.

2.5 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.

2.6 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, $\mathcal{O}(\Delta x^2)$ 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 Problems with code or Tex Blocks

3.1 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).

3.2 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.

3.3 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).

3.4 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.

3.5 Verbatim code blocks inside lists look ugly

Read the Section **??** above. Start the `!bc` and `!ec` tags in column 1 of the file, and be careful with indenting the surrounding plain text of the list item correctly. If you cannot resolve the problem this way, get rid of the list and use paragraph headings instead. In fact, that is what is recommended: avoid verbatim code blocks inside lists (it makes life easier).

3.6 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.

4 Problems with reST/Sphinx Output

4.1 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.

4.2 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.

4.3 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.

4.4 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.

4.5 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.

4.6 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 Problems with LaTeX Output

5.1 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.

5.2 How can I have unnumbered sections?

Use the `--section_numbering=off` option:

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

5.3 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.

5.4 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`.

5.5 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.

5.6 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
```

5.7 How can I use my fancy LaTeX environments?

See Section **??** for how to make a custom theorem environment also in DocOnce, without using implementations restricted to LaTeX.

5.8 The LaTeX file does not compile

If the problem is undefined control sequence involving

```
\Verb!...!
```

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.

5.9 The LaTeX Beamer file does not compile

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

5.10 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.

5.11 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.)

5.12 Chapters are ignored

The default LaTeX style is "article". If you chapters in the DocOnce file, you need to run `ptex2tex` with the option `-DBOOK` to set the LaTeX documentstyle to "book".

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

The default preamble generated by DocOnce has several `preprocess` options to set the font, type of titlepage, etc., when running `ptex2tex` or `doconce ptex2tex`. However, 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.

Note also that `doconce replace` and `doconce subst` can be used after `doconce formt` to tune the LaTeX code in the preamble. There are comment lines with `--- begin preamble ---` and `--- end preamble ---` in the generate `.p.tex` file that can be used to replace the whole preamble by something more suitable.

5.14 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 the `shadedquoteBlue` 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
```

6 Problems with gwiki Output

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 Problems with HTML Output

7.1 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`

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

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 `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:

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

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.

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.

7.3 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: 50%">' mydoc.html
```

8 Problems with reveal/deck HTML5 slides

8.1 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.

8.2 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.

8.3 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.

8.4 Online Python Tutor does not work

Seems to be a problem with reveal slides. The *Forward* button is not clickable.