DocOnce Tutorial: Document Once, Include Anywhere

Hans Petter Langtangen[1,2]

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

Apr 14, 2021

1 Some DocOnce Features

- Strong support for texts with much math and code.
- Same source can produce a variety of output formats:
  - traditional LaTeX B/W documents for printing
  - color LaTeX PDF documents
  - color LaTeX PDF documents for viewing on small phones
  - Sphinx HTML documents with 20+ different designs
  - Plain HTML, Boostrap HTML, or with a template, or with another template, or solarized HTML
  - HTML for Google or Wordpress blog posts
  - MediaWiki (Wikipedia, Wikibooks, etc)
  - Markdown for further processing by Pandoc to a lot of formats
  - Jupyter notebook
  - Other formats include plain untagged text (for email), Creole wiki (for Bitbucket wikis), Google wiki (for Googlecode), and reStructuredText.
- There is an exercise environment with many advanced features such as subexercises, hints and solutions. The document can be compiled with or without solutions.

- With a preprocessor, Preprocess or Mako, one can include other documents (files), large portions of text can be defined in or out of the text. Mako enables use of variables, functions, if-tests, and loops to parameterize the text and generate various versions of the text for different purposes.
- Computer code can be copied directly from parts of source code files.
- Figures and movies with captions, simple tables, URLs with links, index list, labels and references are supported. YouTube and Vimeo videos are automatically embedded in web documents.
- Running text can quickly be edited to slide formats (reveal.js and deck.js, based on HTML5+CSS3).
- Good support for admonitions in various LaTeX and HTML styles for warnings, questions, hints, remarks, summaries, etc.

2 What Does DocOnce Look Like?

DocOnce text looks like ordinary text (much like Markdown[1]), but there are some almost invisible text constructions that allow you to control the formating. Here are some examples.

- Bullet lists automatically arise from lines starting with `*`, or `o` if the list is to be enumerated.
- *Emphasized words* are surrounded by `*`. **Words in boldface** are surrounded by underscores.
- Words from computer code are enclosed in backticks and then typeset `verbatim (in a monospace font)`.
- Section headings are recognized by special decorating characters = before and after the heading, e.g., `===== Here is a subsection heading =====`.
- The abstract of a document starts with *Abstract* as paragraph heading, and all text up to the next heading makes up the abstract,
- Blocks of computer code are surrounded by `!bc` (begin code) and `!ec` (end code) tags on separate lines. Blocks of computer code can also be imported from source files.

[1] In fact, DocOnce allows basic GitHub-extended Markdown syntax as input. This is attractive for newcomers from Markdown or writers who also write Markdown documents (or uses Markdown frequently at GitHub).

- Blocks of LaTeX mathematics are surrounded by `!bt` (begin TeX) and `!et` (end TeX) tags on separate lines. Inline mathematics is surrounded by dollar signs: `$a=b$`.
- Comment lines start with # and are not visible in the output document.
- Comments to authors can be inserted throughout the text with the following syntax and made visible or invisible as desired.

```
[name: comment...]
```

- Title, author, date, figures, and movies are written on one line, starting with a keyword: `TITLE:`, `AUTHOR:`, `DATE:`, `FIGURE:`, and `MOVIE:`.

Write DocOnce documents in a text editor with monospace font!

Some DocOnce constructions are sensitive to whitespace (indentation in lists is a primary example), so you *must* use a text editor with monospace font (also known as verbatim text). Never use fonts like Arial or Helvetica. Other popular markup languages such as Sphinx and Markdown are also sensitive to whitespace and require a monospace font.

2.1 What Can DocOnce Be Used For?

LaTeX is ideal for articles, thesis, and books, but the PDF files does not look fresh and modern on tablets and phones or big computer screens. For the latter type of media you need HTML-based documents with strong support for nice layouts. Tools like Sphinx, Markdown, or plain HTML with Bootstrap are then more appropriate than LaTeX, but these involve a very different syntax. DocOnce lets you write text in one place and then generate the most appropriate language for the media you want to target. DocOnce also has many extra features for supporting large documents with much code and mathematics, not found in any of other publishing tool.

2.2 Writing Guidelines (Especially for LaTeX Users!)

LaTeX writers often have their own writing habits and have preferred LaTeX packages. DocOnce is a simpler format and corresponds to writing in quite plain LaTeX and making the ascii text look nice (be careful with the use of white space!). This means that although DocOnce has borrowed a lot from LaTeX, there are a few points LaTeX writers should pay attention to. Experience shows that these points are so important that we list them *before* we list typical DocOnce syntax!

Any LaTeX syntax in mathematical formulas is accepted when DocOnce translates the text to LaTeX, but the following rules should be followed when translating the text to `sphinx`, `pandoc`, `mwiki`, `html`, or `ipynb` formats.

- AMS LaTeX mathematics is supported, also for the `html`, `sphinx`, and `ipynb` formats.
- If you want LaTeX math blocks to work with `latex`, `html`, `sphinx`, `markdown`, and `ipynb`, only use the following equation environments: `\[ ... \]`, `equation*`, `equation`, `align*`, `align`. `alignat*`, `alignat`. Other environments, such as `split`, `multiline`, `gather` are supported in modern MathJax in HTML and Sphinx, but may have rendering problems (to a larger extent than `equation` and `align`). DocOnce performs extensions to `sphinx`, `ipynb`, and other formats such that labels in `align` and `alignat` environments work well. If you face problems with fancy LaTeX equation environments in web formats, try rewriting with plain `align`, `nonumber`, etc.
- Do not use comments inside equations.
- Newcommands in mathematical formulas are allowed, but not in the running text. Newcommands must be defined in files with names `newcommands*.tex`. Use `\newcommands` and not `\def`. Each newcommand must be defined on a single line. Use Mako functions if you need macros in the running text.
- Use labels and refer to them for sections, figures, movies, and equations only. MediaWiki (`mwiki`) does not support references to equations.
- Spaces are not allowed in labels.
- There is just one `ref` command (no `\eqref` for equations) and references to equations must use parentheses. Never use the tilde (non-breaking space) character before references to figures, sections, etc., but tilde is allowed for references to equations.
- Never use `\pageref` as pages are not a concept in web documents (there is only a `ref` command in DocOnce and it refers to labels).
- Only figures and movies are floating elements in DocOnce, all other elements (code, tables, algorithms) must appear *inline* without numbers or labels for reference[2] (refer to inline elements by a section label). The reason is that floating elements are in general not used in web documents, but we made an exception with figures and movies.

[2] There is an exception: by using *user-defined environments* within `!bu-name` and `!eu-name` directives, it is possible to label any type of text and refer to it. For example, one can have environments for examples, tables, code snippets, theorems, lemmas, etc. One can also use Mako functions to implement environments.

- Keep figure captions shorts as they are used as references in the Sphinx format. Avoid inline mathematics since Sphinx will strip it away in the figure reference. (Many writing styles encourage rich captions that explains everything about the figure; this work well only in the HTML and LaTeX formats.)
- You cannot use `subfigure` to combine several image files in one figure, but you can combine the files into one file using the `doconce combine_images` tool. Refer to individual image files in the caption or text by (e.g.) "left" and "right", or "upper left", "lower right", etc.
- Footnotes can be used as usual in LaTeX, but some HTML formats are not able to display mathematics or inline verbatim or other formatted code (emphasis, boldface, color) in footnotes - keep that in mind.
- Use plain `cite` for references (e.g., `\citeauthor` has no counterpart in DocOnce). The bibliography must be prepared in the Publish format, but import from (clean) BibTeX is possible.
- Use `idx` for index entries, but put the definitions between paragraphs, not inside them (required by Sphinx).
- Use the `\bm` command (from the `bm` package, always included by DocOnce) for boldface in mathematics.
- Make sure all ordinary text starts in column 1 on each line. Equations can be indented. The `\begin{}` and `\end{}` directives should start in column 1.
- If you depend on various LaTeX environments for your writings, you have to give these up, or implement *user-defined environments* in DocOnce. For instance, examples are normally typeset as subsections in DocOnce, but can also utilize a user-defined example environment. Learn about the exercise support in DocOnce for typesetting exercises, problems, and projects.
- Learn about the preprocessors Preprocess and Mako - these are smart tools for, e.g., commenting out/in large portions of text and creating macros.
- Use *generalized references* when referring to companion documents that may later become part of this document (or migrated out of this document).
- Follow recommendations for DocOnce books if you plan to write a book.

Use the preprocessor to tailor output.

If you really need special LaTeX constructs in the LaTeX output from DocOnce, you may use use preprocessor if-tests on the format (typically `#if FORMAT in ("latex", "pdflatex")`) to include such special LaTeX code. With an else clause you can easily create corresponding constructions for other formats. This way of using Preprocess or Mako allows you to use advanced LaTeX features (or HTML features for the HTML formats) to fine tune the resulting document. More tuning can be done by automatic editing of the output file (e.g., `.tex` or `.html`) produced by DocOnce using your own scripts or the `doconce replace` and `doconce subst` commands.

Autotranslation of LaTeX to DocOnce?

The tool `doconce latex2doconce` may help you translate LaTeX files to DocOnce syntax. However, if you use computer code in floating list environments, special packages for typesetting algorithms, example environments, `subfigure` in figures, or a lot of newcommands in the running text, there will be need for a lot of manual edits and adjustments.

For examples, figure environments can only be translated by `doconce latex2doconce` if the label is inside the caption and the figure is typeset like

```
\begin{figure}
  \centering
  \includegraphics[width=0.55\linewidth]{figs/myfig.pdf}
  \caption{This is a figure. \labe{myfig}}
\end{figure}
```

If the LaTeX text is consistent with respect to the placement of the label, a simple script can autoedit the label inside the caption, but many LaTeX writers put the label at different places in different figures, and then it becomes more difficult to autoedit figures and translate them to the DocOnce `FIGURE:` syntax.

Tables are hard to interpret and translate because headings and caption can be typeset in many different ways. The type of table that is recognized looks like

```
\begin{table}
\caption{Here goes the caption.}
\begin{tabular}{lr}
\hline
\multicolumn{1}{c}{$v_0$} & \multicolumn{1}{c}{$f_R(v_0)$}\\
\hline
1.2 & 4.2\\
1.1 & 4.0\\
0.9 & 3.7
```

```
\hline
\end{tabular}
\end{table}
```

Recall that table captions do not make sense in DocOnce since tables must be inlined and explained in the surrounding text.

Footnotes are also problematic for `doconce latex2doconce` since DocOnce footnotes must have the explanation outside the paragraph where the footnote is used. This calls for manual work. The translator from LaTeX to DocOnce will insert `_PROBLEM_` and mark footnotes. One solution is to avoid footnotes in the LaTeX document if fully automatic translation is desired.

2.3 Basic Syntax

Here is an example of some simple text written in the DocOnce format:

```
======= First a Section Heading =======

===== Then a Subsection Heading =====

=== Finally a Subsubection Heading ===

You can also have paragraphs with a bolded paragraph heading. This is
achieved by surrounding the heading with double underscores at the beginning of a line.

__This is a paragraph heading.__
And here comes the text.

===== A Subsection with Sample Text =====
label{my:first:sec}

Ordinary text looks like ordinary text, but must always start at the
beginning of lines. Tags used for _boldface_ words, *emphasized*
words, and `computer` words look natural in plain text.  Quotations
appear inside double backticks and double single quotes, as in ``this
example''.

Below the section title we have a *label*, which can be used to
refer to Section ref{my:first:sec}.
References to equations, such as (ref{myeq1}), work in the same
LaTeX-inspired way.

Lists are typeset as you would do in email,

  * item 1
  * item 2,
    perhaps with a 2nd line
  * item 3

Note the consistent use of indentation (as in Python programming!).
Lists can also have automatically numbered items instead of bullets,

  o item 1
```

```
  o item 2
  o item 3,
    but be careful with the indentation of the next lines!

__Hyperlinks.__
URLs with a link word are possible, as in "hpl": "http://folk.uio.no/hpl".
If the word is just URL, the URL itself becomes the link name,
as in URL: "tutorial.do.txt". DocOnce distinguishes between paper
and screen output. In in PDF generated from LaTeX
generated from DocOnce, the URLs of links appear as footnotes.
With screen output, all links are clickable hyperlinks, except in
the plain text format which does not support hyperlinks.

__Inline comments.__
DocOnce also allows inline comments of the form [name: comment] (with
a space after `name:`), e.g., such as [hpl: here I will make some
remarks to the text]. Inline comments can be removed from the output
by a command-line argument (see Section ref{doconce2formats} for an
example). Inline comments can also be used for detailed editing of
text, much like track changes in word, to illustrate how a text
is revised. (However, for seeing how others have revised the text, I
strongly recommend using Git for version control and running `git diff`
on the appropriate versions, or you can click on differences at
GitHub if the files are hosted there.)

__Footnotes.__ Adding a footnote[^footnote] is also possible.

[^footnote]: The syntax for footnotes is borrowed from Extended Markdown.

__Tables.__
Tables are also written in the plain text way, e.g.,

  |--c--------c-----------c--------|
  |time  | velocity | acceleration |
  |---r-------r-----------r--------|
  | 0.0  | 1.4186   | -5.01        |
  | 2.0  | 1.376512 | 11.919       |
  | 4.0  | 1.1E+1   | 14.717624    |
  |--------------------------------|

The characters `c`, `r`, and `l` can be inserted, as illustrated above,
for aligning the headings and the columns (center, right, left).
One can also use `X` for potentially very long text in a column (will be
left-adjusted).

# Lines beginning with # are comment lines.
```

The DocOnce text above results in the following little document:

3 First a Section Heading

3.1 Then a Subsection Heading

Finally a Subsubection Heading. You can also have paragraphs with a paragraph heading surrounded by double underscores are the beginning of a line.

This is a paragraph heading. And here comes the text.

3.2 A Subsection with Sample Text

Ordinary text looks like ordinary text, but must always start at the beginning of lines. Tags used for **boldface** words, *emphasized* words, and `computer` words look natural in plain text. Quotations appear inside double backticks and double single quotes, as in "this example".

Below the section title we have a *label*, which can be used to refer to Section 3.2. References to equations, such as '(1)', work in the same LaTeX-inspired way.

Lists are typeset as you would do in email,

- item 1
- item 2, perhaps with a 2nd line
- item 3

Note the consistent use of indentation (as in Python programming!). Lists can also have automatically numbered items instead of bullets,

1. item 1
2. item 2
3. item 3, but be careful with the indentation of the next lines!

Hyperlinks. URLs with a link word are possible, as in hpl. If the word is just URL, the URL itself becomes the link name, as in `tutorial.do.txt`. DocOnce distinguishes between paper and screen output. In traditional paper output, PDF generated from LaTeX and DocOnce, the URLs of links appear as footnotes. With screen output, all links are clickable hyperlinks, except in the plain text format which does not support hyperlinks.

Inline comments. DocOnce also allows inline comments of the form **name 1**: comment (with a space after `name:`), e.g., such as **hpl 2**: here I will make some remarks to the text . Inline comments can be removed from the output by a command-line argument (see Section 4 for an example). Inline comments can also be used for detailed editing of text, much like track changes in word, to illustrate how a text is revised. (However, for seeing how others have revised the text, I strongly recommend using Git for version control and running `git diff` on the appropriate versions, or you can click on differences at GitHub if the files are hosted there.)

Footnotes. Adding a footnote[3] is also possible.

Tables. Tables are also written in the plain text way, e.g.,

time	velocity	acceleration
0.0	1.4186	-5.01
2.0	1.376512	11.919
4.0	1.1E+1	14.717624

The characters `c`, `r`, and `l` can be inserted, as illustrated above, for aligning the headings and the columns (center, right, left).

3.3 Mathematics

Inline mathematics, such as $\nu = \sin(x)$ is written exactly as in LaTeX:

```
$\nu = \sin(x)$
```

Blocks of mathematics are typeset with raw LaTeX, inside `!bt` and `!et` (begin TeX, end TeX) directives:

```
!bt
\begin{align}
{\partial u\over\partial t} &= \nabla^2 u + f,
label{myeq1}\\
{\partial v\over\partial t} &= \nabla\cdot(q(u)\nabla v) + g
\end{align}
!et
```

The result looks like this:

$$\frac{\partial u}{\partial t} = \nabla^2 u + f, \tag{1}$$

$$\frac{\partial v}{\partial t} = \nabla\cdot(q(u)\nabla v) + g \tag{2}$$

Of course, such blocks only look nice in formats with support for LaTeX mathematics (this includes `latex`, `pdflatex`, `html`, `sphinx`, `ipynb`, `pandoc`, and `mwiki`). Simpler formats have to just list the raw LaTeX syntax.

Remark.

Although DocOnce allows user-defined styles in the preamble of LaTeX output, HTML-based output cannot make use of such styles. If-else constructs for the preprocessor can be used to allow special LaTeX environments for LaTeX output and alternative typesetting for other formats, but it is recommended to stay away from special environments in the text and write in a simpler fashion. For example, DocOnce has no special con-

[3] The syntax for footnotes is borrowed from Extended Markdown.

struction for algorithms, so these must be simulated by lists or verbatim blocks. Other constructions that should be avoided include margin notes, special tables, and `subfigure` (combine image files to one file instead, via `doconce combine_images`).

3.4 Computer Code

You can have blocks of computer code, starting and ending with `!bc` and `!ec` instructions, respectively.

```
!bc pycod
from math import sin, pi
def myfunc(x):
    return sin(pi*x)

import integrate
I = integrate.trapezoidal(myfunc, 0, pi, 100)
!ec
```

Such blocks are formatted as

```
from math import sin, pi
def myfunc(x):
    return sin(pi*x)

import integrate
I = integrate.trapezoidal(myfunc, 0, pi, 100)
```

A code block must come after some plain sentence (at least for successful output to `sphinx`, `rst`, and formats close to plain text), not directly after a section/paragraph heading or a table.

Blocks of computer code have *named environments*, such as *pycod*. The *py* stands for Python and *cod* indicates a code snippet that cannot be run without more code. Another example is *fpro*, *f* for Fortran and *pro* for a complete program that will run as it stands. There is support for code in C, C++, Fortran, Java, Python, Perl, Ruby, JavaScript, HTML, and LaTeX.

One can also copy computer code directly from files, either the complete file or specified parts, e.g,

```
@@@CODE src/myprog.py fromto: def regression\(@import mymod
```

The copying is based on regular expressions and not on line numbers, which makes the specifications much more robust during software and document developing. With the `@@@CODE` command, computer code is never duplicated in the documentation (important for the principle of avoiding copying information!) and once the software is updated, the next compilation of the document is up-to-date.

Inclusion of files. Another DocOnce document (or any file) can be included by writing `# #include "mynote.do.txt"` at the beginning of a line. DocOnce documents have a `do.txt` extension. The `do` part stands for `doconce`, while the trailing `.txt` denotes a text document so that editors gives you plain text editing capabilities.

3.5 Macros (Newcommands), Cross-References, Index, and Bibliography

DocOnce supports a type of macros via a LaTeX-style *newcommand* construction. The newcommands are defined in files with names `newcommands*.tex`, using standard LaTeX syntax. Only newcommands for use inside LaTeX math environments are supported. (But you can define any type of macros through Mako functions!)

Labels, corss-references, citations, and support of an index and bibliography are much inspired by LaTeX syntax, but DocOnce features no backslashes. Use labels for sections and equations only, and preceed the reference by "Section" or "Chapter", or in case of an equation, surround the reference by parenthesis.

Here is an example:

```
===== My Section =====
label{sec:mysec}

idx{key equation} idx{$\u$ conservation}

We refer to Section ref{sec:yoursec} for background material on
the *key equation*. Here we focus on the extension

# \Ddt, \u and \mycommand are defined in newcommands_keep.tex

!bt
\begin{equation}
\Ddt{\u} = \mycommand{v},
label{mysec:eq:Dudt}
\end{equation}
!et
where $\Ddt{\u}$ is the material derivative of $\u$.
Equation (ref{mysec:eq:Dudt}) is important in a number
of contexts, see cite{Larsen_et_al_2002,Johnson_Friedman_2010a}.
Also, cite{Miller_2000} supports such a view.

As seen in Figure ref{mysec:fig:myfig}, the key equation
features large, smooth regions *and* abrupt changes.

FIGURE: [fig/myfile, width=600 frac=0.9] My figure. label{mysec:fig:myfig}

===== References =====

BIBFILE: papers.pub
```

DocOnce applies Publish for specifying bibliographies because this tool has more functionality than BibTeX, but any BibTeX database can be automatically converted to the simple Publish format.

For further details on functionality and syntax we refer to the DocOnce manual.

4 From DocOnce to Other Formats

We refer to the manual for detailed information on how to compile a DocOnce document to various formats. Here we just give a glimpse of the possibilities.

4.1 Example: Make an HTML File

Suppose you have some DocOnce text in `mydoc.do.txt`. Here is how you compile that document to an HTML file `mydoc.html`, which can be viewed in a web browser:

```
Terminal> doconce format html mydoc --html_style=bootswatch_journal
```

There are lots of styles for HTML files, and `bootswatch_journal` is a fancy one. There are also lots of other command-line options for tailoring the compilation. Run `doconce format -help` to see a list of all options. Those that start with `--html_` are specific for the HTML output format.

4.2 Preprocessors

A DocOnce compilation has three stages:

1. The preprocessor Preprocess is applied to `mydoc.do.txt`, resulting in `tmp_preprocess__mydoc.do.txt`.
2. The preprocessor Mako is applied to `tmp_preprocess__mydoc.do.txt`, resulting in `tmp_mako__mydoc.do.txt`.
3. The text in `tmp_mako__mydoc.do.txt` is translated to the chosen output format.

The preprocessor stages are only run if you have applied preprocessor syntax. The Preprocess program allows you to include other files (usually other DocOnce files) in your document (nested includes are possible). You can also have if-else branching based on variables set on the command line. The Mako preprocessor is (much) more advanced and features if-else tests, loops, variables, and function calls. You can, e.g., write Python functions in Mako to make quite intelligent output (e.g., copy computer code from a certain directory based on a variable that tells which computer language the document is to apply). The preprocessors are definitely one of the strongest features of DocOnce.

4.3 Output Formats

DocOnce can be translated to many formats. For documents with much mathematics and/or computer code the following formats are suitable:

- `latex`, `pdflatex`
- `html`
- `sphinx`
- `ipynb` (IPython/Jupyter notebooks)
- `matlabnb` (Matlab notebooks)

Other formats are

- Wikis: `gwiki` (Googlecode), `cwiki` (Creole), `mwiki` (MediaWiki)
- `plain` (pure ascii)
- `pandoc` (various Markdown formats)

4.4 Slides

DocOnce has strong support for writing slides, see the slides demo for examples. Each slide starts with a subsection heading (5 =), preceded by `!split` to indicate a new slide. Section headings are used to mark parts of the presentation. The slides are compiled as any other DocOnce file, but there is usually a second step where the text is modified to become a proper slide text in the chosen output format. We refer to the manual for details and the DocOnce slide demo.

Several popular slide formats are supported:

- LaTeX Beamer
- HTML5: reveal.js, deck.js, CSSS
- Markdown: Remark
- Plain HTML

5 Demos

Our short scientific report is a good starting point to see how DocOnce documents are written and the vast choice of output formats and settings that are available.

There is also a demo of different slide formats.

DocOnce has support for *responsive* HTML documents with design and functionality based on Bootstrap styles. A Bootstrap demo illustrates the many possibilities for colors and layouts.

DocOnce also has support for exercises in quiz format. Pure quiz files can be *automatically uploaded* to Kahoot! online quiz games operated through smart phones (with the aid of quiztools for DocOnce to Kahoot! translation).

The current text is generated from a DocOnce format stored in the file

```
doc/tutorial/tutorial.do.txt
```

The file `make.sh` in the `tutorial` directory of the DocOnce source code contains a demo of how to produce a variety of formats.