Prepared by Anne Taormina and Sam Fearn, Department of Mathematical Sciences, Durham —
V.1.6 — 13/08/2020

Towards Fully eAccessible Documents

In this document we give an overview of how lecture notes, or other teaching related documents, may be enhanced for increased accessibility. As well as the clear importance of all our students being able to engage with your material as fully as possible, new legislation (see here - https://www.legislation.gov.uk/uksi/2018/952/made) also requires that all material published on University webpages (including on DUO) be made accessible. Making documents which contain substantial amounts of mathematical content fully accessible is challenging, and neither of us would consider ourselves experts on this topic, but here we suggest some steps that we think should be suitable for all colleagues.

We assume that for most colleagues, your lecture notes are currently written up in LaTeX, and that you use pdflatex to produce a pdf file from the source .tex file. There are two main things you can do to enhance accessibility and which we explain in this document.

The first is to take a couple of very simple steps to enhance the accessibility of the pdf file produced by pdflatex. The details behind the accessibility of pdf documents are described here - https://www.homepages.ucl.ac.uk/~ucahmto/elearning/latex/2019/05/06/accessibility-regulations.html. We cover this in the first section of this document.

Although these steps are simple, we also advise producing a copy of your lecture notes in an alternative format, and the government recommends that this be in html (see here - https://www.gov.uk/guidance/how-to-publish-on-gov-uk/accessible-pdfs). This is because the screen readers that many visually impaired users may be using to try to access your notes are typically designed around reading html content rather than pdf. In the second part of this document we describe two alternative methods for automatically producing html content from your source .tex file.

We do not touch upon conversion of a source .tex file into a Microsoft Word document although that format might also be valued by some of our students. Pandoc is in principle capable of converting to this format, but initial results are less impressive than for html. We have found the results similarly lacklustre when using alternative tools for converting to Word format. We therefore think that such conversion is beyond what should be expected in the short term.

From LaTeX to pdf

1. Assuming you have notes typed in LaTeX2e, from which you produce a pdf file, the first thing you should do is visit http://checkers.eiii.eu/en/pdfcheck/ and submit your pdf to check how many of the eAccessibility (European Internet Inclusion Initiative or EIII) tests it passes.

   At first you will only pass some of the 8 tests. Provided your document has a table of contents, you are likely to pass 5 of them.

   Applied Tests

   - (✗ 1) Running Headers and Footers (Test status: Experimental)
   - (✗ 1) Natural Language
   - (✗ 1) Document Title
   - (✓ 1) Correct Tab and Reading Order

- (✔ 1) Bookmarks
- (✔ 1) Scanned Document
- (✔ 1) Structure Elements (tags)
- (✔ 1) Document Permissions

2. By adding

```
\usepackage[pdftex,
        pdftitle={Title of your file},
        pdflang={en-GB}]{hyperref}
```

in the preamble of your LaTeX file, you should pass all tests but the first one (Running Headers and Footers). Just ignore this for the moment.

3. You will pass most of the EIII tests above even if your font is serif. We know sans serif fonts are usually more eAccessible. It is easy to change fonts in a LaTeX file by adding

```
\renewcommand{\familydefault}{\sfdefault}
```

in the preamble.

4. There are also some packages which can be added to your LaTeX file to enhance the accessibility of the resulting pdf files. These have all been recommended to us by colleagues from other institutions with more experience in writing accessible documents. We therefore suggest loading the `hyperref`, `axessibility` and `accessibility` packages:

```
\usepackage{hyperref}
\usepackage{axessibility}
\usepackage{accessibility}
```

If you have an up-to-date installation of LaTeX, you will probably find these packages are all already available on your system. If not, you may need to update your LaTeX installation, or the installed packages.

The `hyperref` package adds hyperlinks throughout your document, but in particular to the references throughout your document, as well as to the table of contents. These hyperlinks make it easier for all users to navigate around your document.

The `axessibility` package adds the literal latex code used in your equations into invisible sections next to each displayed equation. This doesn't change what most users will see in your document, but gives those using screen readers another way of engaging with the intended content. This obviously requires these users to be able to understand latex, but some users may find this the easiest way to access the content.

The `accessibility` package adds invisible tags to your pdf, which may help screen readers understand how to navigate the pdf.

Important remark: It is relatively straightforward to make pdf files pass the EIII tests mentioned above, but although `hyperref` and `axessibility` may improve things somewhat there's no low-effort way to produce genuinely accessible pdf files from

LaTeX. It may not even be possible to produce pdf files with LaTeX that meet the new regulations because of the issues with tagged pdf.

So the above steps provide a first step towards genuine eAccessibility, as far as pdf outputs are concerned. In addition to the above, in order to provide fully accessible documents, you should also make your lecture notes available in html format as we now describe.

From LaTeX to html

There are several ways to convert a LaTeX file to an html output, some slightly easier than others, and which may produce differing results due to the different ways in which they work. As a rule of thumb, existing software (freely available) works best if the LaTeX code is not `too fancy'. We have experimented with notes written by Anne in LaTeX (`Fourier.tex`), which contains figures and equations that are cross-referenced and where theorem, definition, example, remark environments as well as figures are used. The remainder of this section is written with the MacOS user in mind. You may need to adapt the process slightly if you are a Linux user.

After reading the notes below, you might want to run the conversion process for the file `Fourier.tex`. This file and the accompanying figures are available to download from github.com/annetaormina/Latex-code-and-images by clicking on the green `code' button and choosing `download zip' (or use git to clone the repository if you already have git installed). After unzipping the files, you will note that the 8 figures are in a directory called `Figs`, which should be a subdirectory of the directory hosting `Fourier.tex` on your computer. Also note that the preamble in the `Fourier.tex` file instructs you to comment/uncomment certain actions depending on whether you plan to use LaTeXML or pandoc to convert to html. The `Fourier.tex` version in github.com/annetaormina/Latex-code-and-images has the instructions relevant for pandoc commented out.

Anne - LaTeXML

After experimenting with Rstudio (Rmarkdown, bookdown) and notably pandoc (see the next subsection), I settled for LaTeXML which worked *almost* effortlessly with the `Fourier.tex` file I used as test file. Those who are not MacOS users might find the website https://dlmf.nist.gov/LaTeXML/get.html useful to get LaTeXML on their preferred platform.

Instructions to install LaTeXML on MacOS

I installed LaTeX on my MAC (OS Catalina version 10.15.5) via Homebrew (https://brew.sh/) by following the excellent tutorial at https://www.youtube.com/watch?v=SELYgZvAZbU. Homebrew is a package manager for MacOS.

First one must install Homebrew. This is done in two steps from a terminal window.

1. At the prompt in your terminal window, type

```
xcode-select --install
```

This installs command line tools needed to install Homebrew. ***Note:*** *These tools may have been installed previously on your computer, in which case the computer will return*

```
xcode-select: error: command line tools are already installed,
use "Software Update" to install updates
```

Unless you wish to update the software, you can ignore this error and proceed to install Homebrew.

2. Now you are ready to install Homebrew. At the prompt in your terminal window, type

```
/bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/
            Homebrew/install/master/install.sh)"
```

This command is given on the homepage of Homebrew at https://brew.sh, under the words `Install Homebrew'. As soon as you have typed the script above, your computer will start the download and ask for your superuser (root) password, which you should enter. During the download and installation, you will be asked to

```
Press RETURN to continue or any other key to abort
```

so press return. If all goes well, after a few minutes, Homebrew will be installed. You should see the words `Installation successful' at some point.

A few useful brew commands lines

- `brew help`
- `brew search`

  (lists all packages you might want to download with Homebrew; there are several thousands). If you know the package you wish to download, check if it is downloadable via Homebrew by typing

- `brew search packagename`
- `brew install packagename`
- `brew info packagename`

  (gives where and when you downloaded the package as well as more analytic data)

- `which packagename`

  (this gives the location of the package, but this command might give a symbolic link like /usr/local/bin/packagename. In this case enter

  `ls -la/usr/local/bin/packagename`

  to see the actual path (for instance `usr/local/Cellar/packagename/…`)

With Homebrew operational on your computer, you can install a variety of software packages with minimal know-how with command lines on a terminal window.

To install the package LaTeXML, simply type

```
brew install latexml
```

at the prompt in your terminal window. The installing time should be quite short.

You are now ready to convert LaTeX documents to html documents using LaTeXML.

LaTeXML instructions

Let `file.tex` be the LaTeX file you wish to convert into an html document using LaTeXML, and let `Directory` be the name of the directory in which `file.tex` sits.

1. If you use figures, you should place them in `Directory` as well (or in a subdirectory of `Directory` if you prefer). It is preferable to use the .jpg format for figures and images as they convert effortlessly through LaTeXML.

2. With LaTeXML installed, open a terminal window and ensure you are in `Directory`.

   Run the command lines

   ```
   latexml --dest=output.xml --includestyles file.tex
   ```

   and then

   ```
   latexmlpost --dest=file.html output.xml
   ```

   (the option `--includestyles` in the first command has been added but you will not need it unless you add to `Directory` some style files from extra packages you have to install for your particular LaTeX code).

3. Once `file.html` has been generated, edit the file and add the following code lines in-between `<head>` and `</head>` to load MathJax and ensure that MathML is rendered in a variety of browsers (not only Firefox, but Safari, Chrome etc):

   just after <head>, insert the script

   ```
   <script src="https://cdnjs.cloudflare.com/ajax/libs/mathjax/
2.7.3/MathJax.js?config=TeX-MML-AM_CHTML"></script>
   ```

   and just before </head>, insert

   ```
   <link rel="stylesheet" href="https://samfearn.github.io/
latexml.min.css">
   ```

4. Save `file.html`, which is by default in `Directory`, alongside the class files `LaTeXML.css` and `ltx-article.css` which are generated in the process of running LaTeXML. Note, by default the styles in these class files are ignored in the html (please contact Sam if you want to know the details here).

5. open `file.html` in your favourite browser and compare it with the pdf output `file.pdf` generated with `pdflatex`. You should aim at having the rendering of `file.html` as similar as possible to `file.pdf`. The class file `latexml.min.css` has been tailored to achieve this, but depending on the complexity of your LaTeX code, you may have to find bespoke solutions. These could

involve modifying your LaTeX code slightly or create an additional local css file, which will have to be uploaded on DUO alongside your html file following the instructions given in the last section of this document, on `sharing the resulting html'.

Note that each time you run LaTeXML on `file.tex`, the `file.html` output has no memory of the edits you made in item 3. So you will have to redo points 3 and 4 before comparing again with `file.pdf`. There is a slicker way to proceed, which involves writing a bash script to execute all the commands under items 3 and 4 without you having to edit `file.html` at all. This bash script must be located in `Directory` as well.

Aside: to create a bash script, open the MAC application TextEdit, click file -> new then click Format -> Make plain text. A simple bash script is provided by typing the following two lines in the plain text document:

```
#!/bin/bash

echo Hello World
```

Save the file under a name of your choice, say `dotest` (make sure no extension .txt is provided). Now locate the file using Finder and right click on it and go to `get info' from the drop down menu. On the bottom right window that opens with the information relating to your file, there is a padlock. If it is locked, click on it and provide your superuser password to unlock the file. The next step is to make `dotest` an executable file. From a terminal window, navigate to the folder in which `dotest` is located, then at the prompt type the command line

```
chmod 700 dotest
```

You now have an executable bash script. To check it works, type the command line

```
./dotest
```

in your terminal window, and you should see the words `Hello World' appear on the next line.

Let `dofile` be the name of the bash script relevant to the conversion from `file.tex` to `file.html`. Then `dofile` must *at least* contain the following instructions:

```
#!/bin/bash

FILE=$PWD/"file"


/usr/local/bin/latexml --dest=$FILE.xml --includestyles $FILE.tex


/usr/local/bin/latexmlpost --dest=$FILE.html $FILE.xml


sed -i ' ' '/<head>/ a\

       <script src=\"https:\/\/cdnjs.cloudflare.com/ajax/libs/
mathjax/2.7.3/MathJax.js?config=TeX-MML-AM_CHTML\"><\/script>'
$FILE.html
```

```
sed -i ' ' '/<\/head>/ i\

    <link rel="stylesheet" href=\"https:\/\/samfearn.github.io\/
latexml.min.css\">' $FILE.html
```

Note that the instructions above are specific to MacOS. One might want to add extra instructions, relating to a given LaTeX code. I provide the bash script `doFourier.sh` relevant to `Fourier.tex`, at github.com/annetaormina/Latex-code-and-images. It is easy to adapt it for your own `file.tex` by replacing Fourier by the name of your LaTex file in the first executable line. The bash script `doFourier.sh` has an extra line giving specific instructions about the padding of some frames that appear in `Fourier.tex`.

***Final Remark*:** It is likely that some of the macros you regularly use in LaTeX will not translate properly into html. In such cases, maybe it pays to reconsider the use of these macros and see if you can find a work around. For instance the environment `\fbox{ \begin{minipage} …\end{minipage}}` does not export itself through LaTeXml, so it might be better to avoid this environment (it works for pdf outputs, so you could create a pdf output with that box around your minipage, but then you would have to change your LaTeX code if you wish to export as html. In `Fourier.tex` , I have used a new environment called Framed, defined in the preamble of `Fourier.tex`, which allows the use of the align environment within a box).

Sam - Pandoc

My preferred method is to use a tool called Pandoc, available for Windows, MacOS and Linux. On Mac, this can either be installed by downloading an install package, or by using Homebrew as Anne describes above. One of the advantages of pandoc over LaTeXML is that pandoc has the capability to convert between more formats than just tex to html, and can be customised more easily than LaTeXML.

Like LaTeXML, pandoc is a command line tool. However, there does exist a graphical front-end for pandoc called PanWriter. I haven't tested this myself, but I believe it is reasonably popular and well developed - this tool is also available for Windows, Mac and Linux. In the rest of this section I'll be assuming pandoc is installed and being used from the command line; I'll assume the file you wish to convert is called `file.tex`.

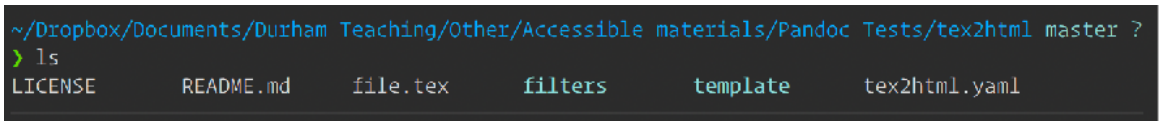
At its most basic, once pandoc has been installed `file.tex` can be converted to html using the following command:

```
pandoc file.tex -s --mathjax -o file_converted.html
```

Notice that we use the '`-s`' option to tell pandoc to produce a *standalone* file, which is ready to be opened by a web browser. If this option isn't used, pandoc produces a snippet — this is like having the tex commands for a file, without putting in `\documentclass{article}`, `\begin{document}`, etc. We also tell pandoc that we'd like the html file to load the mathjax library.

The default output of pandoc can be improved with a few simple additions - namely expanding macros defined in your tex, loading a stylesheet in the output html to make this output look better (and be more responsive on mobile devices), and support for some additional latex environments.

To make this as simple as possible, I've created a project on GitHub which contains the required configuration changes and additional scripts to use with pandoc. These files are available on my github here - https://github.com/samfearn/tex2html. To use these additions, you can download a zip of the files from the previous link (click the green 'Code' button, then 'download ZIP', or use git to clone the repository if you already have git installed). Once you download and unzip the files from github, you should have a folder called `latex2html-master` (or just `latex2html` if using git clone). These files need to be moved from this folder and placed alongside your `file.tex`.



Pandoc can then be run using the command:

```
pandoc file.tex -d tex2html
```

This uses a default filename defined in the `tex2html.yaml` configuration file. If you'd prefer to specify the output filename explicitly (as say file.html) you can instead use the command

```
pandoc file.tex -d tex2html -o file.html
```

There is an option in the configuration file to specify the name of a folder relative to `file.tex` if you like to store your images in a subfolder. Here you will also find options as to which level (section, subsection…) to base environment numbers from, as well as which environments share the same counter, the possibility to load additional css and other options.

The following environments are styled automatically in the output html: `\begin{theorem}`, `\begin{lemma}`, `\begin{definition}`, `\begin{proof}`, `\begin{example}`, `\begin{framed}`, `\begin{proposition}`, `\begin{center}`, `\begin{remark}`. In order for this to work, the following need to be added to the top of `file.tex` before it is converted:

```
\newenvironment{lemma}
      {\begin{verse}
      Lemma:
      }
      {\end{verse}}
\newenvironment{theorem}
      {\begin{verse}
      Theorem:
      }
      {\end{verse}}
\newenvironment{definition}
      {\begin{verse}
```

```
     Definition:
     }
     {\end{verse}}
\newenvironment{proof}
     {\begin{verse}
     Proof:
     }
     {\end{verse}}

newenvironment{example}
     {\begin{verse}
     Example:
     }
     {\end{verse}}
newenvironment{framed}
     {\begin{verse}
     Framed:
     }
     {\end{verse}}
\newenvironment{proposition}
     {\begin{verse}
     Proposition:
     }
     {\end{verse}}
\renewenvironment{center}
     {\begin{verse}
     Center:
     }
     {\end{verse}}
\newenvironment{remark}
     {\begin{verse}
```

```
Remark:

}

{\end{verse}}
```

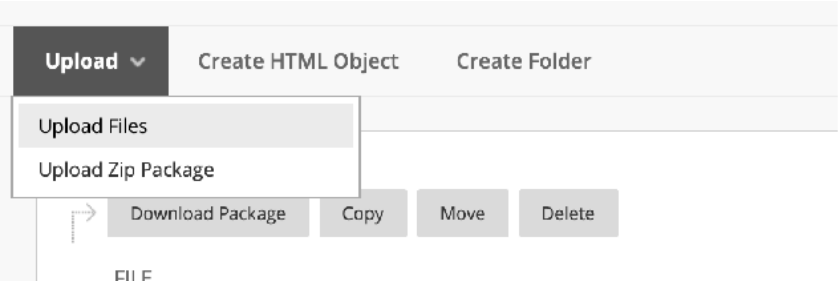
These environment definitions only need to be present when using pandoc to convert to html, if you run pdflatex after adding these definitions, you may find you pdf file looks different to what was expected. Simply comment out these lines in the .tex file if you want to run pdflatex. If you have already defined environments with these names, you should comment out your old definitions when adding these new ones to use Pandoc (or, if you make sure these new definitions occur later in the file than the previous definitions, just change the relevant commands to `\renewenvironment` in the above).

Sharing the resulting html

Your University/Department may provide you with some public html space that can be used to share the resulting html with your students (this is the case for the Department of Mathematical Sciences in Durham). If this is the case, you can simply copy the final `file.html` to the relevant folder on the web server, alongside any required images in their corresponding locations (a `Figs/` subfolder in the example available from github.com/annetaormina/Latex-code-and-images), and provide a link to the new file.

You may instead wish to share the html through your Virtual Learning Environment (VLE), which in the case of Durham is Blackboard, in order to limit the availability of your notes to your students and potentially, Durham colleagues in your Department. The following instructions describe how to make the html available in Blackboard under Durham's configuration, though the instructions for other instances of Blackboard, or other VLEs should be similar. For the remainder of this section we refer to Blackboard as configured at Durham as DUO.

First, navigate to the DUO course you wish to add the notes to. In the 'Course Management' menu, available in the left-hand sidebar, click the 'Files' heading, and then click on the top option in the list, which should be for the files specific to the relevant DUO course. At the top of the main panel, you should then see a button titled 'Upload', which opens a new menu when you hover your cursor over the button, and you should click 'Upload Files' in this submenu:

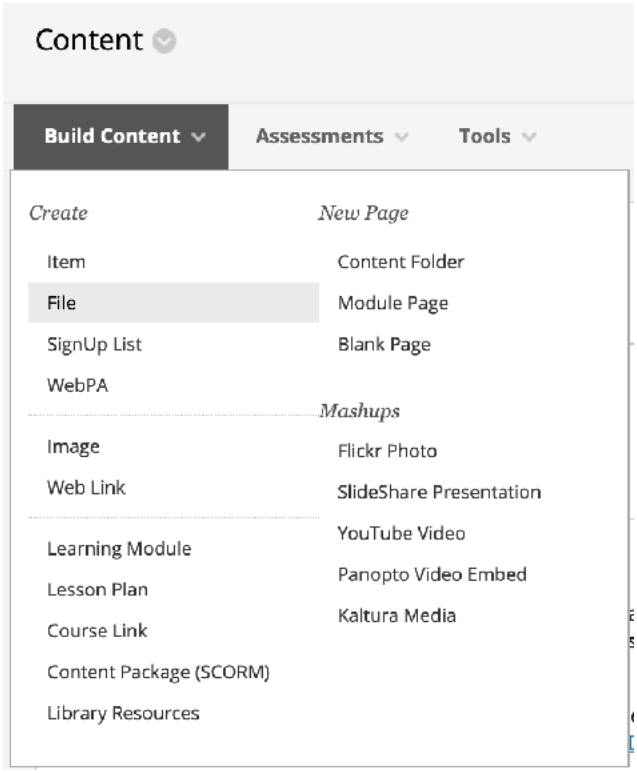


Once you click the 'Upload Files' button, you will be presented with a form to select the files you wish to upload. Click 'Browse My Computer', and then select the html file you wish to upload. Finally, click Submit at the bottom of the form to upload the html file.

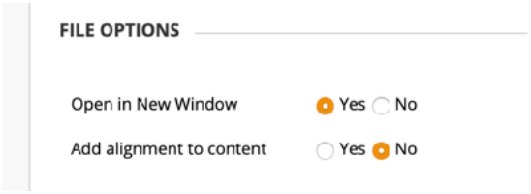
In order for your html file to see any required images, they will need to be put in a folder with the same name as on your computer - `Figs/` in the example from before. To create a new folder, click the 'Create Folder' button from the menu at the top of the main files page (visible in the above screenshot), give the folder the appropriate name (`Figs` unless you changed this for your own notes) and click 'Submit'. You then need to click on your new folder in the files menu, in order to open this folder and upload the images

inside this folder. Once you've clicked on the new folder, you can use the 'Upload Files' button as before to upload all your image files into this new folder.

Now you've added the html and any required images into DUO, the last step is to make a link to this available for the students. To do so, navigate to a page you can add content to, make sure edit mode is turned on, and click 'Build Content' from the top menu, then select 'File' from the submenu which appears:



Give the content item an appropriate name, such as 'Lecture Notes', and then click on 'Browse Content Collection'. This will open a new panel allowing you to select the html file you uploaded before. You need to click in the radio box at the far left of the html file in order to select it, clicking the file name will simply open the file in a new tab or window. Click submit at the bottom of the form. You will then be presented with a form with options. You should change the option 'Open in New Window' to 'Yes'.



Finally, assuming the remaining options are suitable for your needs, click 'Submit' at the bottom of the form. You now have a content item which opens your html in a new tab (or window, depending on browser settings) when clicked.

If you wish to modify the presentation of the html, by customising the css, you will need to modify your html to load a local css file with a relative file path, and then upload your additional css to the relevant relative location compared to your html using the DUO files upload.

Conclusion

This document is very much `work in progress', and we welcome comments/suggestions that could make the whole process much smoother. If you have any query or question about eAccessibility and the provision of eAccessible documents, feel free to contact one of us and we will try to help (s.m.fearn@durham.ac.uk, anne.taormina@durham.ac.uk).