An example of the ‘accessibility’ style file in use

Andy Clifton

June 23, 2019

Abstract

Structured and tagged PDFs are required to meet modern corporate and governmental standards for document accessibility. PDFs that are created with core LaTeX are not tagged or structured, making it difficult to use LaTeX in a corporate or government environment. This document explains how LaTeX can be used to prepare documents that pass such tests.

This document is intended to be used as a test case as it contains most of the elements of a technical LaTeX document, including horrific formatting, custom fonts, complex document structures, lists, equations, figures and code listings.

Table of Contents

List of Figures

List of Tables

1 Introduction

The de-facto standard for scientific publishing is LaTeX. LaTeX is often preferred over WYSIWYG word processors for technical documents because of the relatively simple file format that can be shared across users on many different platforms, and the ease of formatting a document for journal publication.

However, one issue with using LaTeX is document *accessibility*. Accessibility is important for documents produced by federally-funded organizations: since the US Congress passed the 1998 Section 508 Amendment to the Rehabilitation Act of 1973, it has been a requirement that all federally-funded documents are accessible to people with disabilities.

An accessible PDF has several characteristics:

- All of the document content has been tagged
- It is possible to define a reading order based on those tags
- Images and links are given alternate text descriptions
- Tables are tagged, so that the table structure can be established
- Unicode descriptions of all characters are required

A document that has these characteristics is often referred to as being ‘508 compliant’. As 508-compliance is often judged using automated tests on the *.pdf* file, there is no option to work around this by using careful text descriptions of figures, for example.

In this document, I explain how LaTeXcan be generated using the ***accessibility*** style file.

My goal is that this will be a ‘living’ document and template that can be updated as we gain new insight into this process.

2 Some more text

Table 1 lists the packages that are included in this demonstration article. These packages often call other packages, so this is not an exhaustive list.

2.1 Accessibility support

LaTeX does not prepare a structured PDF document directly. Instead, we use the ***accessibility*** package to do this for us. This generates a tagged PDF that passes most automated document tests.

2.1.1 Alternative text

Alternative text, or 'Alt text', is a textual description of an equation, link or figure that can be used to replace the visual information in that element. This is often seen as a text 'pop-up' in PDF readers. Alt text can be added after the PDF is compiled using a PDF editor such as Adobe's Acrobat Pro. Alternatively – and probably best for ensuring that the final document is what the author intended – it can be generated from within the source document using the **`pdftooltip`** environment from the ***pdfcomment*** package.

For example, Figure 1 has been labeled with a tool tip.

Note that the `subfig` and `subfigure` packages are deprecated and so subfigures are implemented using the `subcaption` package. The `subcaption` package appears to be the most frequently maintained package at this time, and contains the same functionality as the `subfig` and `subfigure` packages.

As a further demonstration that tooltips actually work, passing the pointer over the following equation should reveal a pop-up:

$$a^2 + b^2 = c^2 \tag{1}$$

The ***accessibility*** package includes an `\alt{}` environment which is intended to create a tool tip. Although it has been included in the source of the next equation, it does not currently work.

$$a^2 + b^2 = c^2 \tag{2}$$

2.1.2 Problems with embedded fonts

One requirement of passing automated tests for accessibility is that fonts must be embedded in the the final PDF. You can check the PDF for embedded fonts using a PDF viewer. For example, in Adobe Acrobat Reader, look at the 'fonts' tag of the document properties. If any fonts are not shown as being an *embedded subset*, you need to try again.

Encapsulated postscript figures are particularly prone to having undefined fonts. Check by compiling your document in draft mode, and seeing if the fonts are still present in the output PDF. To fix this problem, you could consider changing the *.eps* file to a *.png*. If you wish to do this 'on the fly', you could use this approach in your preamble:

2.2 Including code listings

The ***listings*** package is one of several packages that can be used to typeset source code, and is used in this document. It seems to work.

3 A template

The code used to produce this document is available from https://github.com/AndyClifton/accessibility.

4 Problems with this approach

Well, there are lots. If you find any, please use GitHub's issue tracking to report these. You can find the current list of issues at `https://github.com/AndyClifton/accessibility/issues`.

5 Conclusions

A LaTeX style file was created in order to generate accessible *.pdf* files with LaTeXÀccessible *.pdf* files are compiled directly from the LaTeXsource code.

Acknowledgements

This document benefitted from contributions to the website, `http://tex.stackexchange.com/`.

Babett Schalitz produced the original ***accessibility*** package in 2007. That package was oriented towards KOMA-script documents. It was not accepted by CTAN and was subsequently not available to the LaTeXcommunity.

Babett Schalitz provided me with a copy of the original ***accessibility*** package in May 2019 and asked me to take

Table 1. Packages explicitly loaded for this document

Packages	options	functionality
nag		checks that packages are up to date and looks for bad habits in LaTeX code.
geometry		sets page size and margins
mathptmx		changes fonts
helvet		changes fonts
courier		changes fonts
amsfonts, amssymb		supplies fonts that are useful for mathematics
booktabs		
graphicx		graphics handling, including *.eps* figures (see Section 2.1.1)
natbib	sort	handles citations and allows the `\cite`, `\citep` and `\citet` citation commands.
fontenc	T1	
xcolor		
babel	english	
subcaption		provides the `subfigure` environment to produce sub figures
hyphenat		
setspace		
parskip		
toclof	subfigure	
toclifbind	nottoc, notlot, notlof	
todonotes		inline and margin to-do notes
listings		
caption		
cmap		
pdfcomment		tool-tips. Also calls the package ***hyperref***

(a) A chick.

(b) Another chick

Figure 1. Test images

up maintenance with a goal of submitting it to CTAN. This document is intended to support that effort. I am extremely grateful for all of Babett's work!