LaTeX Tagged PDF project progress report for summer 2024

Frank Mittelbach, Ulrike Fischer

Abstract

The LaTeX Tagged PDF project was started in spring 2020 and announced to the TeX community by the LaTeX team at the (online) 2020 TUG conference. This short report describes some of the progress in this multi-year project made during 2024.

Contents

1 Introduction

For a description of the background, goals and previous progress reports of the LaTeX Tagged PDF project we refer the reader to various previous articles [8, 1, 6, 2]. This report concentrates on a few important additions in the past year.

As a quick summary, the most important goals of the Tagged PDF project are:

- to improve accessibility of PDF documents produced by LaTeX;
- to provide tagging of PDF in an automatic and easy way;
- to also improve conversion to HTML;
- to push the use of PDF 2.0 — better for accessibility, especially if math is involved;
- and to push improvements in viewers and tools.

On all these topics there has been noticeable progress.

2 Why bother?

In his article about "Signing PDF files" [4], Hans Hagen wrote "*Personally I wonder why one would use PDF to provide adaptive accessibility, because HTML is meant for that.*" This is a sentiment that you encounter quite often: That PDF is such a bad format that it is not worth the time to improve its accessibility.

There is clearly some truth in this. Nobody can deny that HTML is more accessible. It is a structured language, so the structure is there from the start and every viewer that wants to render an HTML page has to understand this structure. Also HTML has a long history of accessibility support, with various

doi.org/10.47397/tb/45-2/tb140mittelbach-tagging24

well-known and well-understood standards and implementations. PDF, on the other hand, is a page description language where structure can be added *optionally* (by "tagging" the PDF) and as the structure is not relevant for rendering, its use by viewers is optional too. While there is a PDF standard for accessibility, PDF/UA, it is not really well understood how conforming processors should behave, and it is not easy to test if a PDF is accessible or not. So why do we bother?

The answer is twofold. Firstly, while HTML is a nice format for accessibility it has also some drawbacks. For instance, HTML is not a single file. HTML pages can load graphics, CSS files, JavaScript files, webfonts, session cookies and more from many places and servers. This makes it difficult to store an HTML "document" for offline reading or for archiving. You never know if tomorrow you will see the same thing if you reload the HTML — and so the general advice for keeping evidence about what's shown on a web page is to make a screenshot or to print into a PDF. The large number of files also makes it difficult to forward or distribute a text in HTML format. In this respect PDF is clearly the better format as it is a single self-contained file and works offline without problems; with PDF/A, a well-known and well-understood standard for archivable PDFs exists.

Furthermore, HTML is not intended to guarantee a faithful and unchanging representation. Instead it deliberately delegates some of the visual presentation decisions to the browser that can adjust the interpretation of the HTML structure to outer circumstances or consumer choices, e.g., the window size, the available fonts, etc. In contrast, PDF provides such a faithful visual presentation of the document, capturing the exact intentions of its author, regardless of the printer or viewer used to render the PDF.

Both models have their important use cases and so it is not surprising that PDF has been widely used for more than thirty years and in all likelihood will continue to be so used when controlled presentation form is important. Thus, PDF is also highly important for users with special accessibility requirement, and the fact that most PDF documents are essentially not accessible is a major concern, which we address with this project.

The second reason is to improve and simplify HTML production. Currently all existing workflows that create HTML from LaTeX sources are based on patching and overwriting package code. For example, `tex4ht` contains many `.4ht` files (such as `biblatex.4ht` and `enumitem.4ht`). Analogously, the `latexml` workflow contains many so-called "bindings" (`biblatex.sty.ltxml`, `enumitem.sty.ltxml`), and the `lwarp` package also (`lwarp-biblatex.sty`, `lwarp-enumitem.sty`). All these files do at their core is essentially the same: they reconfigure LaTeX commands and environments so that they produce a structure suitable for HTML. All this configuration work is done without direct contact with the package authors. Thus, if a package changes or extends it features, or if a new package appears on CTAN, all HTML converter have to adapt their configuration files individually and on their own. The Tagged PDF project is different here: while it also changes LaTeX commands and environments so that they produce a structure, the goal is to make changes in the kernel and in the packages directly. Once the structure is there, it can than also be used to create HTML without the need of many fragile external patches.

3 The WTPDF (Well Tagged PDF) examples

At the begin of 2024 two standards for PDF 2.0 were finally released: PDF/UA-2, the ISO standard for Universal Accessibility in PDF 2.0 [5] and WTPDF (Well Tagged PDF) for Accessibility and Reuse in PDF 2.0 [10]. The members of the PDF association were asked for examples and the LaTeX team was one of the first to provide a reasonably large set of more or less randomly chosen texts. The set covers a variety of document types and demonstrates various tagging techniques — and also open problems. E.g., the Bible, a document with simple tagging but with many structures due to the large number of verses, pushed hard on some limits (it can not be compiled with pdfLaTeX) and revealed a number of bugs that slowed down both compilation and validation. The Max and Moritz example demonstrates problems in the handling of documents with more than one language: we were not yet able yet to convince the speech reader to use the correct voice when switching from one language to another. The amsmath documentation, `amsldoc`, demonstrates the current state of math tagging.

These examples and many more, along with their sources, can be found at `github.com/latex3/tagging-project/discussions/72`.

4 Tagging status of LaTeX packages and classes

One step of the project is the adaptation of external packages and classes. For this we have created a database that shows the tagging status of various packages and classes.[1] A week before the TUG conference, the database contained around 200 entries; since then it has grown at great speed and now covers over 1000 packages and classes. The database shows if a package is compatible, partially compatible or currently incompatible with the tagging code and links to issues and test files. If we do not have tests to certify the status it is listed as unknown.

The database is meant as help both for users who want to know if a package can be safely used and for package developers, who can check the status of the packages they maintain.

The database is a `yaml` file. A subset of around 700 entries can be viewed on this web page: `latex3.github.io/tagging-project/tagging-status`.

5 Progress in math tagging

Accessibility of math in a PDF is clearly not very good. In PDF/UA-1 and PDF 1.7 or earlier there is basically no provision for good math tagging — the best one can do is to add some alternative text and hope that the PDF reader doesn't mess up punctuation, etc. In PDF 2.0 there are better options: it supports the MathML namespace and allows to attach additional files (e.g., a MathML representation or the LaTeX source of an equation) to the math structure, but these new options are not well supported by consumer applications. The creation and publishing of the WTPDF examples mentioned above triggered some development here. For example, the next release of the Foxit PDF reader [3] will extract a MathML file and pass it on to the AT-technology. Together with changes in the NVDA screen reader [9] it will greatly improve the reading of math. More details can be found in [7]. We expect that other applications will follow, now that there are a growing number of real documents to support.

[1] We thank here Ian Thompson who created the initial list and Matthew Bertucci who greatly extended and improved it, providing many test files for future use.

References

[1] U. Fischer. On the road to Tagged PDF: About StructElem, marked content, PDF/A and squeezed Bärs. *TUGboat* 42(2):170–173, 2021. `doi.org/10.47397/tb/42-2/tb131fischer-tagpdf`

[2] U. Fischer, F. Mittelbach. Automated tagging of LaTeX documents — what is possible today, in 2023? *TUGboat* 44(2):262–266, 2023. `doi.org/10.47397/tb/44-2/tb137fischer-tagging23`

[3] Foxit. PDF reader. `www.foxit.com/pdf-reader/`

[4] H. Hagen. Signing PDF files. *TUGboat* 45(1):145–149, 2024. `doi.org/10.47397/tb/45-1/tb139hagen-pdfsign`

[5] *ISO/FDIS 14289-2; Document management applications — Electronic document file format enhancement for accessibility — Part 2: Use of ISO 32000-2 (PDF/UA-2)*, 1st ed., 2024. `www.iso.org/standard/82278.html`

[6] F. Mittelbach, U. Fischer. The LaTeX Tagged PDF project — a status and progress report. *TUGboat* 43(3):268–272, 2022. `doi.org/10.47397/tb/43-3/tb135mitt-tagged`

[7] F. Mittelbach, U. Fischer. Enhancing LaTeX to automatically produce tagged and accessible PDF. *TUGboat* 45(1):52–59, 2024. `doi.org/10.47397/tb/45-1/tb139mitt-deims24`

[8] F. Mittelbach, C. Rowley. LaTeX Tagged PDF — a blueprint for a large project. *TUGboat* 41(3):292–298, 2020. `doi.org/10.47397/tb/41-3/tb129mitt-tagpdf`

[9] NV Access. NVDA screenreader. `www.nvaccess.org/download`

[10] PDF Association. *Well-Tagged PDF (WTPDF)*, Feb. 2024. Version 1.0.0. `pdfa.org/wp-content/uploads/2024/02/Well-Tagged-PDF-WTPDF-1.0.pdf`

◇ Frank Mittelbach
Mainz, Germany
`https://www.latex-project.org`

◇ Ulrike Fischer
Bonn, Germany
`https://www.latex-project.org`