Accessible Documents in LaTeX

https://github.com/rhstanton/accessible_LaTeX

Version 2.1.0

Richard Stanton
richard.stanton@berkeley.edu
UC Berkeley

November 14, 2026

If Ally doesn't score this PDF 100%, don't panic — that's expected.

Template v2.0 targets the newer PDF/UA-2 + PDF/A-4 (PDF 2.0) profile. Most accessibility validators were calibrated for the older UA-1 / PDF 1.7 profile and haven't caught up, so **Ally will report a score below 100%**. That reflects checker lag, not real accessibility problems. Ally raises *several false* errors: "The PDF does not have a title" (the title *is* present, in the XMP metadata that PDF 2.0 uses), and "Image without a description" on *every* equation (the math is tagged as MathML, which Ally doesn't check for). The PDF passes veraPDF — the meaningful UA-2 / PDF/A-4 check — cleanly; see "Validator-specific notes" below and `VALIDATION.md`.

Do not accept Ally's offered "fixes" to raise the score — they make the PDF *less* accessible, not more. See "Validator-specific notes" below and `VALIDATION.md`.

1 Introduction

On January 8, 2026, we were notified by campus that, beginning in April 2026 (now May 11):

> "The updated requirements of the ADA require that digital course materials provided to students, even materials inside password-protected course sites like bCourses, will need to comply with accessibility standards (Web Content Accessibility Guidelines (WCAG) 2.1 Level AA)."

Many of us use LaTeX to create teaching materials—both slides and documents. Standard LaTeX (including Beamer) does not automatically generate accessible PDFs.

2 What is this project?

An experiment exploring accessible LaTeX documents using the LaTeX Tagging Project:

- Two document classes: articles (`article`) and slides (`ltx-talk`).
- Two flavors of each: a minimal template (`minimal_slides.tex`, `minimal_article.tex`, ∼40 lines each, runnable in a minute) plus a fully annotated template (`accessible_slides.tex`, `accessible_article.tex`) that doubles as documentation.
- Shows: what's common to both, what's different.
- Contains: working examples with math, text, graphics, and tables.

- Validation: passes veraPDF (PDF/A-4 + UA-2). Heads up: popular validators (especially Ally) report *false positives* under PDF/UA-2 – not bugs in the templates. See "Validator-specific notes" below, and `VALIDATION.md` in the repo

2.1 How to use it

1. **Learn the common requirements:** Apply to any LaTeX document
2. **See class-specific needs:** Understand slides vs. articles
3. **Copy and adapt:** Use as templates for your documents
4. **Study the code:** Heavily commented for learning

Available at: https://github.com/rhstanton/accessible_LaTeX

2.2 Converting to accessible LaTeX: Overview

- **Your existing LaTeX skills transfer** - you're just adding accessibility features
- **The changes are minimal** and follow consistent patterns

Common requirements (both slides & articles):

- Add document metadata and enable PDF tagging
- Tag images with alt text; tag table headers
- Use accessible colors; compile with LuaLaTeX

Key difference:

- Articles: Keep your existing class! Just add accessibility features
- Slides: Switch to `ltx-talk` (similar syntax, but recreate styling)

Both require LaTeX kernel 2025-11-01 or later (see next sections for details).

See "Getting started" sections at the end for detailed steps. For complete LaTeX Tagging Project documentation, see latest detailed instructions.

3 TeX version requirements (both slides and articles)

- Minimum: TeX Live 2023 or later with all packages updated
- Critical: Must have LaTeX kernel 2025-11-01 or later
  - Update via TeX Live Manager (Windows) or TeX Live Utility (Mac)
  - Or use Overleaf Labs (see next section)
- Will NOT work: TeX Live 2022 or earlier

4 You CAN use Overleaf (both slides and articles)

- `ltx-talk` requires a very recent **LaTeX kernel** (2025-11-01 or later)
- This is available through Overleaf's Labs program (not standard Overleaf)

4.1 Using Overleaf (2 steps)

1. **Join Overleaf Labs**:
   - Visit https://www.overleaf.com/labs/participate
   - Opt in and enable "Rolling TeX Live releases"
2. **Configure project**:
   - Set TeX Live version to "Rolling TeXLive (labs)" (bottom of list)
   - Set Compiler to LuaLaTeX

Resources: https://docs.overleaf.com/writing-and-editing/creating-accessible-pdfs

5 Common requirements for ALL documents

Whether you're creating slides or articles, these requirements are mostly the same:

5.1 Document Metadata

- Every accessible document must begin with `\DocumentMetadata`
- Goes *before* `\documentclass`

```
\DocumentMetadata{
   pdfstandard=a-4,       % PDF/A-4 archival standard (no embedded files)
   pdfstandard=ua-2,      % Add PDF/UA-2 conformance target
   lang=en-US,               % Language for screen readers
   tagging=on,            % Enable PDF tagging
   tagging-setup={
      math/setup=mathml-SE % Keep math accessibility enabled (semantic MathML)
   }
}
```

This configures accessibility, math tagging, and validator-compatibility behavior in one place.
This boxed example shows the current metadata profile used in this article template.
The `tagging-setup` entries keep math accessible while reducing helper attachments that can trigger warnings in some strict conformance validators.

An Ally finding we deliberately leave unfixed. Blackboard Ally reports every inline equation as "image without description." Under PDF/UA-2 the kernel attaches *MathML* (not `/Alt`) to each `Formula` tag, and that MathML is what carries the math semantics to a screen reader; Ally simply checks for `/Alt` and has no PDF/UA-2 support yet, so it misreports the formula as undescribed. **Do not silence this with `\tagpdfsetup{math/alt/use}`** (earlier versions of this template did). That attaches the raw LaTeX source as `/Alt`, and per PDF/UA a screen reader reads an element's `/Alt` *instead of* descending into its children — so the `/Alt` *hides* the MathML, and the user hears the verbatim LaTeX read aloud instead of the rendered equation. Treat the Ally finding as a false positive. See `VALIDATION.md`.

5.2 Images, Tables, and Colors

- **Images:** All images must include alt text descriptions
  - Allows screen readers to describe visual content
  - Even decorative images need marking
  - See dedicated "Figures" section for examples

- **Tables:** Must specify which rows are headers
  - Helps screen readers navigate table structure
  - Essential for data accessibility
  - See dedicated "Tables" section for examples

- **Accessible colors:** WCAG 2.1 requires 4.5:1 contrast ratio

```
\colorlet{AccessibleRed}{red!80!black}
\colorlet{AccessibleGreen}{green!40!black}
% Standard blue is fine as-is
```

5.3 LuaLaTeX Compilation

- You must compile with LuaLaTeX (not pdfLaTeX or XeLaTeX)

- **Why LuaLaTeX?**
  1. **Automatic MathML:** Makes math accessible without manual work
  2. **Full tagging support:** Complete PDF accessibility features
  3. **Modern fonts:** Handles Unicode and OpenType fonts

- **How to switch:**
  - Command line: `lualatex filename.tex`
  - Most editors: Select "LuaLaTeX" from compiler menu

6 The basics

- Use standard LaTeX environments: `section`, `subsection`, `itemize`, `enumerate`, etc.
- **Existing source files don't need a lot of editing**
- Here's some gratuitous *math* for the accessibility checker

7 Figures

When including a figure, you must provide alt text describing the image for screen readers. This is *required* for accessibility compliance.

```
\includegraphics[height=.4\textheight,alt={A capybara's head facing left}]{capybara.jpg}
```

7.1 Caption vs. alt text: they describe different things

A common mistake is to put the same words in `\caption{...}` and `alt={...}`. They have different audiences:

- **Caption** (visible to everyone): an editorial title or annotation — what this figure is being used for in the surrounding argument. Sighted readers see the image *and* the caption.
- **Alt text** (read by screen readers in place of the image): a literal description of what's in the image. A screen-reader user is hearing this *instead of* seeing the picture, so it should stand alone.

In the example below the caption says why the figure is here ("a non-decorative illustration"), while the alt text describes what is actually depicted ("A capybara's head facing left"). For a purely decorative image, use `alt={}` so the screen reader skips it.



Figure 1: A capybara, included here as a non-decorative illustration of accessible figure markup.

8 Tables

When including a table, you must specify which rows are headers. Screen readers use this information to help users navigate table content. Use `\tagpdfsetup{table/header-rows={...}}` *before* the `tabular` environment:

- Use `{1}` for 1 header row
- Use `{1,2}` for 2 header rows
- Use `{1,2,3}` for 3 header rows, etc.
- **Validator note:** PAC can report `Table header cell has no associated subcells` even when headers are tagged correctly; the warning is still emitted by PAC 26.1.0 (March 2026). See PAC release notes at https://pac.pdf-accessibility.org/en/resources/release-notes.

8.1 Example: Table with 3 Header Rows

```
\tagpdfsetup{table/header-rows={1,2,3}}
\begin{tabular}{ccccrcccr}
\toprule
← 3 header rows
\midrule
← data rows
\bottomrule
\end{tabular}
```

Payment date	Caplet expiry date	DF_{pay}	Forward rate	Days to expiry	Days in accrual period	T_{expiry}	Δ	Caplet
2004/12/01	—	0.99550	0.01790	0	91	0.00000	0.25278	—
2005/03/01	2004/11/29	0.99008	0.02188	89	90	0.24384	0.25000	1,178.77
2005/06/01	2005/02/25	0.98401	0.02413	177	92	0.48493	0.25556	4,844.73
2005/09/01	2005/05/27	0.97733	0.02675	268	92	0.73425	0.25556	10,016.71

Table 1: A table

9 Code listings

For code, use a real verbatim environment rather than `\texttt` inside boxes; the block is then tagged as text and read aloud by screen readers.

Tagging caveat. As of TeX Live 2026 the two most popular code-listing packages, `listings` and `minted`, are *not yet compatible* with the LaTeX tagging engine. `listings` is tracked at https://github.com/latex3/tagging-project/issue `minted` hits the same problem because it loads `fvextra`, tracked at https://github.com/latex3/tagging-project/issues/1060. Both produce visually pleasing syntax-highlighted output but emit untagged graphical artifacts that screen readers skip.

The working alternative today is the `fancyvrb` package – specifically the `Verbatim` environment and the `\VerbatimInput` command, which is what `ltx-talk`'s own demos use. No syntax highlighting (yet), but the code is tagged and screen-readable, which is the accessibility-first tradeoff:

When `listings`/`minted` become tagging-compatible upstream, you can switch back; the configuration block near the top of this file shows the WCAG-AA-safe color palette to use.

10 Common pitfalls

- Confusing PDF tagging with PDF bookmarks
  - `tagging=on` creates structure tree for screen readers (required for accessibility!)
  - PDF bookmarks (navigation outline in left pane) are separate and optional
  - Bookmarks do NOT generate automatically from `tagging=on`
  - For articles: use `\tableofcontents` or the `bookmark` package
  - For slides: See custom bookmark code in `accessible_slides.tex` preamble (Part 2)
- Forgetting alt text for images
  - Every `\includegraphics` needs an `alt={...}` parameter
  - For purely decorative images, use empty alt text: `alt={}`

- Not specifying table header rows
  - Add `\tagpdfsetup{table/header-rows={...}}` before each table
  - Use `{1}` for 1 header row, `{1,2}` for 2 header rows, etc.
- Insufficient color contrast
  - WCAG 2.1 requires 4.5:1 contrast ratio for normal text
  - Avoid light colors: `yellow`, `cyan` fail contrast requirements
  - Darken `red` and `green`: use `red!80!black`, `green!40!black`
  - Standard `blue` is fine and meets WCAG requirements
  - Test with a contrast checker: https://webaim.org/resources/contrastchecker/
- Using the wrong compiler
  - Make sure your editor is set to use LuaLaTeX, not pdfLaTeX
- Old TeX distribution
  - TeX Live 2022 or earlier won't work
  - Update packages using TeX Live Manager (Windows) or TeX Live Utility (Mac)

11 What to ADD to existing documents

When converting existing LaTeX files to be accessible, here's what you need to add:
Note: This section shows the current profile used by this article template.

11.1 Required additions for ALL documents

1. **At the very top** (before `\documentclass`):

```
\DocumentMetadata{
   pdfstandard=a-4,
   pdfstandard=ua-2,
   lang=en-US,
   tagging=on,
   tagging-setup={...see template...}
}
```

2. **In every** `\includegraphics` command:

```
\includegraphics[alt={Description of image}]{file}
```

3. **Before every table**:

```
\tagpdfsetup{table/header-rows={1}}
\begin{tabular}{...}
```

4. **Change your compiler to LuaLaTeX** (not pdfLaTeX or XeLaTeX)

11.2 Additional for existing Beamer slides

- Change `\documentclass{beamer}` to `\documentclass[frame-title-arg]{ltx-talk}`
- Remove all Beamer themes, color schemes, and template commands
- Keep your `\begin{frame}` environments as-is
- Recreate styling using standard LaTeX packages (xcolor, fontspec, etc.)
- This is one-time preamble work—then reuse for all future talks!

12 Getting started: Common steps for both

1. **Setup environment:** Use Overleaf Labs OR install/update TeX Live locally
   - Overleaf: See earlier section for Labs setup
   - Local: Update via TeX Live Manager (Windows) or TeX Live Utility (Mac)
2. **Get templates:** Download from https://github.com/rhstanton/accessible_LaTeX. For a quick first try, start with `minimal_slides.tex` or `minimal_article.tex` (~40 lines each, runnable in a minute); the fully annotated versions add documentation and optional features on top.
3. **Add accessibility features:**
   - Add `alt` text to images
   - Add `table/header-rows` to tables
4. **Set compiler to LuaLaTeX**
5. **Compile with LuaLaTeX**

13 Getting started: What's different by document type

13.1 For Articles (using `article`, `report`, `book`, etc.)

- Add `\DocumentMetadata` before `\documentclass`
- Include the `tagging-setup` block in `\DocumentMetadata` (as shown in this template)
- Switch to `fontspec` and `unicode-math` packages
- Keep your existing documentclass!

13.2 For Slides (migrating from Beamer)

- Copy preamble from `accessible_slides.tex`
- Change `\documentclass{beamer}` to `\documentclass[frame-title-arg]{ltx-talk}`
- Remove Beamer themes/templates/colors
- Recreate styling using standard LaTeX/xcolor
- One-time preamble work—then reuse for all future talks!

14 Validation workflow: use multiple checkers

- No single checker catches everything.
- Accessibility usability checks and strict PDF conformance checks are not identical.
- A practical workflow is to run more than one checker and compare findings.

Checkers used so far: Ally (the accessibility checker built into bCourses), veraPDF (open-source PDF conformance validator, https://verapdf.org/), PAC (PDF Accessibility Checker, https://pac.pdf-accessibility.org/), and Acrobat.

15 Optional validator cleanup step: `strip_af.py`

- The metadata settings produce a fully tagged PDF that passes veraPDF (PDF/A-4 + UA-2). **Heads up:** Ally currently reports one false error under PDF/UA-2 (legacy PDF-1.x title check). See "Validator-specific notes" below.
- Stricter checkers (e.g., veraPDF) may still complain about /AF or embedded-file structures.
- Optional strict-archival cleanup command:

  ```
  python strip_af.py build/yourfile.pdf
  ```

- Then validate `build/yourfile.pdf` with `veraPDF`.
- Prerequisite for the script: `python -m pip install pikepdf`

16 Validator-specific notes

No single accessibility checker catches everything. Treat one-tool warnings as signals to investigate, not as proof of source errors. The items below are known false positives that can be ignored. See `VALIDATION.md` in the repo for the full per-tool status.

16.1 Per-tool quirks (also present under UA-1)

- **Acrobat: "Lbl and LBody – Failed."** Raised on tagged lists, figure captions, table captions, and section numbers even when the `<Lbl>` tag is structurally correct. Known Acrobat checker bug. Reference: https://tex.stackexchange.com/a/709655/2388.
- **PAC: "Table header cell has no associated subcells."** Reported even when `\tagpdfsetup{table/header-rows={…}}` is set correctly. Still present in PAC 26.1.0 (March 2026). Track: https://pac.pdf-accessibility.org/en/resourc

16.2 UA-2 / PDF 2.0 checker-coverage gaps (new since template v2.0)

Template v2.0 upgraded the targets from PDF/UA-1 + PDF/A-2 (PDF 1.7) to PDF/UA-2 + PDF/A-4 (PDF 2.0). UA-2 is the newer, stricter, more accessible standard, but most validators were written for UA-1 and haven't fully caught up. The reports below look like errors; they're checker lag, not template regressions. The meaningful UA-2 conformance check today is veraPDF, which passes both example PDFs.

- **PAC validates against PDF/UA-1, not UA-2.** At open, PAC warns it "currently fully supports documents up to and including PDF 1.7." The test report itself shows `Standard: PDF/UA-1` — PAC has no UA-2 checker and silently falls back. Expect a non-trivial failure count on the test report because UA-2 introduces structure elements and role names the UA-1 checker doesn't recognize. These are not real conformance bugs.

- **Ally: "The PDF does not have a title."** Ally checks the legacy PDF 1.x `/Title` key, which PDF 2.0 deprecates in favor of XMP `dc:title`. LaTeX writes XMP; veraPDF accepts it; Ally has not caught up.

Do NOT accept Ally's offered "fix" for this finding.
It may downgrade the file to PDF 1.7, destroying PDF/UA-2 and PDF/A-4 conformance.

Questions or suggestions? richard.stanton@berkeley.edu