Accessible Slides in LaTeX

https://github.com/rhstanton/accessible_LaTeX

Version 2.1.0

Richard Stanton

richard.stanton@berkeley.edu

UC Berkeley

November 14, 2026

© Richard Stanton, 2026

Don’t worry if Ally doesn’t give you 100%

- **Ally will *not* score these PDFs 100%** — and that’s expected: checker lag, not real accessibility problems.
- Both example PDFs pass veraPDF (strict UA-2 / PDF/A-4) cleanly. veraPDF is the meaningful check, not the Ally score.
- Ally has no PDF/UA-2 support yet, so it raises **several false positives**:
  - “No title” — the title *is* present, in XMP (PDF 2.0).
  - “Image without a description” on *every* equation — the math is tagged as MathML, which Ally doesn’t check.
- PAC falls back to UA-1 and reports a longer list. Details: end of deck + `VALIDATION.md`.

Do NOT accept Ally’s offered “fixes.”

They raise the score but make the PDF *less* accessible, not more.

Why v2.0 upgraded to PDF/UA-2 + PDF/A-4

What changed. v2.0 updates the `\DocumentMetadata{…}` target from the old PDF/UA-1 + PDF/A-2 (PDF 1.7) to PDF/UA-2 + PDF/A-4 (PDF 2.0).

Why this is worth doing:

- PDF/UA-2 (ISO 14289-2, 2024) is the current accessibility standard. It improves tagging for math (MathML), tables, and lists over UA-1, and is the long-term direction for accessible PDF.
- PDF/A-4 (ISO 19005-4) is the current archival standard, layered on PDF 2.0 instead of PDF 1.7. It's cleaner about deprecating legacy metadata and aligns with UA-2.
- Net effect: better screen-reader support, future-proof, and matches what modern LaTeX tagging actually produces.

The tradeoff: most accessibility checkers were written for UA-1 / PDF 1.7.

- Haven't fully caught up with new standard.
- The fix is on the checker side, not the document side.

LaTeX and accessibility: new requirements

- From May 11, 2026, course materials must meet WCAG 2.1 Level AA accessibility standards
  - Applies even to password-protected course sites (e.g., bCourses).
- Many of us use LaTeX to create teaching materials, both slides and documents.
- **But standard LaTeX (especially Beamer) does not automatically generate accessible PDFs.**
- Good news: The LaTeX Project team has developed solutions.

What is this project?

- A practical template set for accessible LaTeX 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, runnable in a minute) plus a fully annotated template (`accessible_slides.tex`, `accessible_article.tex`) that doubles as documentation
  - Contains: math, text, figures, and tables

- Also includes some (optional) additional slide features
  - Readable outline slide
  - Automatic section separator slides
  - Multi-page bibliography support

- Repo: https://github.com/rhstanton/accessible_LaTeX

Converting to accessible LaTeX: the big picture

- Your existing LaTeX skills transfer: mainly adding a few consistent accessibility features.
- Most changes are local: metadata once; then per-figure and per-table tweaks.

- **Applies to both slides & articles:**
  - Document metadata + PDF tagging
  - Alt text on images
  - Table header row tagging
  - Sufficient color contrast
  - Compile with LuaLaTeX

- **Key difference:**
  - Articles: keep your existing class and add tagging features
  - Slides: switch to `ltx-talk` (frame syntax similar to Beamer), then recreate styling

Outline

1. Setup

Version requirements: what actually matters

- Engine: must compile with LuaLaTeX (not pdfLaTeX or XeLaTeX).
- LaTeX kernel/format date: must be 2025-11-01 or later.
- **Important:** a "TeX Live year" alone is not a reliable guarantee.
  - You need a sufficiently new LaTeX kernel *on your install*, or use Overleaf Labs.
- **Quick check (in any document preamble):**

```
\show\fmtversion
```

Overleaf (works **when configured correctly**)

- `ltx-talk` requires a very recent LaTeX kernel (format date 2025-11-01+).
- On Overleaf, this typically means using the Labs program (not standard Overleaf).

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

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

2. Core requirements

Core requirement 1: document metadata

- **Required for both slides and articles**
- Every accessible document must begin with `\DocumentMetadata`
- Goes before `\documentclass`

```
\DocumentMetadata{
   % Request both PDF/A archival conformance and PDF/UA accessibility tagging
   pdfstandard=a-4,         % PDF/A-4 format (use a-4f only if embedding 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    % Enable semantic MathML generation
   }
}
```

- **Why PDF/UA-2 + PDF/A-4?** This aligns with current ISO standards and modern tagging output.
- **Ally quirk (and why we do *not* "fix" it):** Ally flags every inline equation as "image without description."
  - Under PDF/UA-2 the kernel attaches MathML (not `/Alt`) to each `Formula` tag — the MathML carries the math semantics.
  - Ally checks for `/Alt` and reports the formula as undescribed when it is absent.
  - Do not "solve" this with `\tagpdfsetup{math/alt/use}`. That attaches the raw LaTeX source as

Core requirements 2–4: images, tables, and color contrast

- **Required for both slides and articles.**

- **Images:** every `\includegraphics` needs `alt={...}`.
- **Tables:** mark header rows using `\tagpdfsetup{table/header-rows={...}}`.
- **Colors:** WCAG 2.1 AA requires 4.5:1 contrast ratio for normal text.

- **Accessible color defaults:**

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

Core requirement 5: compile with LuaLaTeX

- **Required for both slides and articles.**
- Compile with LuaLaTeX (not pdfLaTeX or XeLaTeX).

- **Why LuaLaTeX?**
  1. Automatic math accessibility support (no manual MathML files in normal workflows)
  2. Full tagging support
  3. Good Unicode/OpenType font support

- **How to switch:**
  - Command line: `lualatex filename.tex`
  - Editors: select `LuaLaTeX` as the compiler

3. Examples

Slide content: mostly the same as Beamer

- Slides live in `frame` environments (similar to Beamer).
- So your content often transfers with minimal changes.

- **Typical frame skeleton:**

```
\begin{frame}{Title}
\begin{itemize}
  \item First point
\end{itemize}
\end{frame}
```

- Note (optional): template fonts keep digits/percent/dollar signs consistent in math/text.
  - Text: $1234567890%.
  - Math: $1234567890%.

Figures: alt text

- Figures are included as usual (e.g., `\includegraphics`), but add alt text.
- **Good (minimal):** `\includegraphics[alt={A capybara}]{capybara.jpg}`
- **Better (more informative):**

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

- Decorative images: use empty alt text: `alt={}`.



Caption vs. alt text: they describe different things

- A common mistake: putting the same words in `\caption{...}` and `alt={...}`.
- Different audiences, different jobs:
  - Caption: visible to *everyone*; an editorial label saying *why* this figure is here.
  - Alt text: read *instead of* the image by screen readers; a literal description of *what is depicted*.

- **Example:**

```
\includegraphics[
   alt={A capybara's head facing left}
]{capybara.jpg}
\caption{A capybara, used as a non-decorative illustration.}
```

- Decorative image ⇒ `alt={}` so screen readers skip it.

Tables: header rows

- Tables use `tabular` as usual, but you must specify header rows.
- Use `{1}` for one header row, `{1,2}` for two header rows, etc.
- **Note:** PAC may still warn about header associations in some cases; see Validator-specific notes.

```
\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

Code listings: contrast and tagging

- For code, use a real verbatim environment, not `\texttt` inside boxes – screen readers then see the code as text.
- Heads up: as of TeX Live 2026, the popular syntax-highlighting packages `listings` and `minted` are not yet compatible with the LaTeX tagging engine; they emit untagged graphical blocks. Tracked at https://github.com/latex3/tagging-project/issues/70 and https://github.com/latex3/tagging-project/issues/1060.
- The temporary working path is `fancyvrb`'s `\VerbatimInput` (used below). Monochrome, but tagged – which is what accessibility needs.

4. Slides versus articles

Articles vs. slides

- **Articles:** keep your existing class (e.g., `article`) and add the checklist items.
- **Slides:** switch to `ltx-talk`.
  - Your `\begin{frame}` content often stays the same.
  - Remove Beamer themes/templates; recreate styling with standard LaTeX packages (as in this template).

- **If you're converting Beamer slides:**

```
\documentclass{ltx-talk}
```
(instead of beamer)

Getting started (new vs. converting): keep it simple

- **New documents:**
  - Try first: `minimal_slides.tex` or `minimal_article.tex` (~40 lines each) — the bare-minimum runnable example.
  - Then copy: `accessible_slides.tex` or `accessible_article.tex` for the fully featured template, replace content, keep the preamble structure.

- **Converting existing documents:**
  - add metadata + LuaLaTeX + alt text + table header rows
  - for slides, switch to `ltx-talk` and redo styling once

- Repo: https://github.com/rhstanton/accessible_LaTeX

5. Validation and known issues

Common pitfalls

- Confusing PDF tagging with PDF bookmarks
  - `tagging=on` creates the structure tree for screen readers (required).
  - PDF bookmarks (navigation pane) are separate and optional.
- Forgetting alt text
  - Every `\includegraphics` needs `alt={...}` (or `alt={}` if decorative).
- Not tagging table header rows
  - Add `\tagpdfsetup{table/header-rows={...}}` before each table.
- Low contrast colors
  - Avoid light colors (e.g., plain yellow/cyan).
  - Use darker variants for red/green (defined in this template).
- Wrong engine or old kernel
  - Use LuaLaTeX and ensure format date `2025-11-01` or later.

Validation workflow: use multiple checkers

- No single checker catches everything.
- Accessibility usability checks and strict PDF conformance checks are not identical.
- Practical workflow: run 2+ tools and compare findings.

- **Common tools:**
  - Ally (bCourses)
  - veraPDF (PDF/A conformance): https://verapdf.org/
  - PAC (PDF accessibility): https://pac.pdf-accessibility.org/
  - Adobe Acrobat (useful but has known quirks)

Optional cleanup for strict validators: `strip_af.py`

- The metadata settings produce a fully tagged PDF. Note: Ally reports one false “missing title” error under PDF/UA-2 – see “Validator-specific notes” for why to ignore it.
- Some strict tools (e.g., veraPDF) may still complain about `/AF` or embedded-file structures.
- To remove these, run:

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

- Prerequisite: install the Python package pikepdf first: `python -m pip install pikepdf`.
- Then validate again with veraPDF.

Validator-specific notes (reference)

New under PDF/UA-2 (already covered earlier):

- Ally reports "The PDF does not have a title" — it reads the legacy PDF-1.x `/Title` key; do not accept its offered "fix".
- PAC 26.1.0 has no UA-2 checker; it silently validates against PDF/UA-1. Failure counts are UA-1 vs UA-2 differences, not real bugs.

Older false positives (also present under UA-1):

- **Acrobat: "Lbl and LBody – Failed."** Known Acrobat checker bug on tagged lists, captions, and section numbers. Reference: https://tex.stackexchange.com/a/709655/2388.
- **PAC: "Table header cell has no associated subcells."** Reported even when `table/header-rows` is set correctly. Still present in PAC 26.1.0. Release notes: https://pac.pdf-accessibility.org/en/resources/release-notes.

Full status and per-tool details: `VALIDATION.md` in the repo.

6. (Optional) additional utilities

Optional presentation utilities

- This template includes optional helpers split across two packages
  - Not required for accessibility
  - Useful for common presentation tasks
  - Easy to customize or remove (each package is independent)

- **Two optional features:**
  1. Automatic section separator slides (in `slide_utils.sty`)
  2. Multi-page bibliography support (in `slide_bib.sty`)

- See those `.sty` files for implementation details and customization options.
- **Note:** There's additional code in the main preamble to get readable table of contents.

Optional feature 1: section separator slides

- **What it does:** Automatically inserts a separator slide after each `\section` command
  - Displays section number and title in large text
  - Centered on the slide for visual emphasis
  - Uses accessible colors (AccessibleRed for sections, blue for subsections)

- **Example:** After `\section{Core requirements}`, automatically creates a slide saying

3. Core requirements

- **To disable:** remove `\usepackage{slide_utils}` or comment out Section 1 of `slide_utils.sty`
- **To customize:** Edit colors, fonts, or spacing in the frame template

Optional feature 2: multi-page bibliography

- **Problem:** `\bibliography` inside `frame` environment puts all references on one slide.
  - For large bibliographies, this can be unreadable.
  - This was easy with Beamer, using the `allowframebreaks` option.
  - But `ltx-talk` does not have a similar option.

- **Solution:** Write code to split bibliography automatically across multiple slides
  - Works with standard BibTeX workflow
  - Uses Lua script (`rsbibsplit.lua`) to split the `.bbl` file
  - (LuaLaTeX provides access to the Lua programming language)

- **Usage:** As usual (see end of this `.tex` file for complete example)
  - `\cite{key1,key2,...}` to specify references
  - `\bibliography{bibfile}` to specify `bib` file
  - (Set `\RS@bibmax` to customize max per slide; default=8).
  - **Note:** Do **not** put `\bibliography` command inside `frame` environment

- See `slide_bib.sty` for detailed workflow explanation.

7. Quick start

Quick start: accessible LaTeX on one slide

- **To make an accessible LaTeX PDF:**
  1. Add `\DocumentMetadata{...}` (before `\documentclass`)
  2. Compile with LuaLaTeX
  3. Every figure: `\includegraphics[alt={...}]{...}`
  4. Every table: `\tagpdfsetup{table/header-rows={...}}`
  5. Use high-contrast colors (4.5:1) and validate with 2+ tools

- Template repo: https://github.com/rhstanton/accessible_LaTeX
- Questions: richard.stanton@berkeley.edu

8. References