# LaTeX News

Issue 44, November 2026 (LaTeX release 2026-11-01)

*Contents*

**Introduction** 2

**News from the Tagged PDF project** 2
- TUG 2026, Calgary . . . 2
- ltx-talk: an accessible class for presentations . . 2
- Tags for paragraphs . . . 2
- Provide an `order` key mechanism for templates 3

**Tagged PDF: theorem-like environments** 3
- Use the `order` key mechanism for theorem-like environments and proofs . . . 3
- Fix a color leakage in theorem-like environments 3
- Subtle improvement for theorem-like captions . 3
- Theorem-like environment instances for AMS classes . . . 4

**Tagged PDF: block and list environments** 4
- Revisiting the para-handling logic for block environments . . . 4
- Replaced `legacy-code` key in blockenv template 4
- `\newlist` failed for `description` lists . . . 4
- Make `\item[]` produce an empty item label . . 4
- Extending the enumitem emulation . . . 4

**Tagged PDF: heading commands** 4
- New tag names for heading structures . . . 4
- Use the `order` key mechanism for heading commands . . . 4
- Support `force-unnumbered` key in heading templates . . . 5
- Support `placement=ragged` in heading templates . . . 5
- Heading templates now support column spanning . . . 5
- Setting left and right margins for (some) headings . . . 5
- Logic of `after-penalty-vspace` key improved 5
- Conditional heading template instances . . . 6
- Replaced `start-code` and `final-code` keys in heading templates . . . 6
- Run-in headings followed by displayed math . . 6
- Heading template instances for AMS classes . . 6
- Emulation of titlesec package started . . . 6
- Tagging of headings in minipages and parboxes 7

**Tagged PDF: other fixes and improvements** 7
- Package fncychap is now tagging compatible . . 7
- Package listings is now (mostly) tagging compatible . . . 7
- Package float is now tagging compatible . . . 7
- Storing more titles for named references . . . 7
- Fixed issue with `end` hooks for math environments . . . 7
- More control over the tagging of title and headings . . . 7

**New or improved commands** 8
- Testing whether a certain template is defined . 8
- New Lua helper `luatexbase.new_luacommand` 8
- Lua callbacks get unique identifiers . . . 8
- Conditional adding of punctuation . . . 8
- Providing `\strongemph` . . . 8
- Supporting alternative template key names . . 8
- Modifying template key values . . . 9

**Code improvements** 9
- Support for `\UseName` in template key bindings 9
- Further removal of "fake math" from the kernel 9
- `\OmitIndent` now usable in vertical mode . . . 9

**Bug fixes** 9
- Avoid problems with page breaks in the middle of verbatim-like environments . . . 9
- Support nested requests for class or package target dates . . . 9

**First aids** 9
- Fix amsbook running headers . . . 9
- First aid for subcaption used with amsmath . . 9

**Changes to packages in the amsmath category** 10
- Drop special handling for LuaTeX in `\genfrac` 10

**Changes to packages in the tools category** 10
- trace: support use with tagging . . . 10
- bm support for (lua-)unicode-math . . . 10
- afterpage could lose pages in some contexts . . 10
- showkeys: avoid double printing . . . 10
- longtable: separate data for unnumbered tables 10

**Documentation changes** 10
- Tagging LaTeX News . . . 10
- US spelling . . . 10

LaTeX News, and the LaTeX software, are brought to you by the LaTeX Project Team; Copyright 2026, license LPPL.

## Introduction

As with the previous releases of LaTeX, our major focus remains on improving the out-of-the-box accessibility of LaTeX; it is gratifying to see how well this now works, and that more and more people are embarking on this journey with us. Not surprisingly, a large portion of this newsletter therefore covers improvements in that space.

Of course, there are also other improvements and corrections, as you will find out if you read on.

## News from the Tagged PDF project

### TUG 2026, Calgary

This year's TUG conference was held in Calgary, Canada. There were many talks related to accessibility, all worth checking out on YouTube [3]. Many of them have been written up in the conference issue of *TUGboat* [4]. Among them are two talks from LaTeX team members: Ulrike's "News from the LaTeX Tagged PDF project: 2026" [5] and Frank's "Towards accessible document classes" [6]. Timothy Prescott gave two presentations: one about "Converting a PDF textbook to be accessible" with the help of the recent LaTeX release [7]; and one about "Creating accessible presentations with ltx-talk" [4, p. 208] describing the recent work by Joseph Wright on ltx-talk that offers a beamer replacement with which one can produce accessible slides. John Lienhard discussed his work on "Modernizing MIT's thesis template: mitthesis.cls" [4, p. 212], again making use of work by the LaTeX team. On the general struggle with non-accessible material, Aarush Bhat (a blind student) and Timothy Erickson, his tutor, gave a talk on "Visual impairment meets mathematics: Bypassing institutional inertia" [4, p. 195]—all these, and many more, are worth listening to or reading.

### ltx-talk: an accessible class for presentations

Early in the planning for the accessible PDF project, we decided that the beamer class would not be able to follow the pattern of "upgrading in-place" that we are using for most documents. Whilst beamer has a (generally) very good *user* interface, the code internals are complex, and the entire class has no tests or comments which would allow adjustment.

Joseph therefore started on a brand-new class for presentations, ltx-talk;[1] work commenced in 2023 but moved slowly, reaching the first testing release last June. The key aims of ltx-talk are to keep the (good) beamer syntax for users, but to write the code from the outset to support tagging.

At the time of writing, the ltx-talk class has reached v0.6.10, with 55 releases in total. Those have seen new features added and a large number of issues addressed, but there is still more to do. The first priority for class development has been to provide suitable logic structure, with design aspects more limited to date. Getting ltx-talk right also needs supporting work in the kernel and the L3 programming layer, but this is all progressing well.

A typical ltx-talk document looks pretty much the same as a beamer one, other than requiring the command `\DocumentMetadata` on the first line:

```
\DocumentMetadata{tagging = on}
\documentclass{ltx-talk}
\begin{document}
\begin{frame}
  \frametitle{Some things we did}
  \begin{itemize}[<+->]
    \item Get cape,
    \item Wear cape,
    \item \dots{}
  \end{itemize}
\end{frame}
\end{document}
```

This will produce three slides, with the list revealed one item at a time, in the visual output, but in the tagging is a single list without repetition.

Some aspects of the input syntax for ltx-talk are still open, most notably whether all frames should have a title or, as at present, this should be given using an optional `\frametitle` command (as shown). However, the majority of the input syntax is sufficiently stable that users can convert their beamer content: see Timothy Prescott's presentation at TUG'26, for example.

Presentation design in ltx-talk is driven by templates, but at present these are very much placeholders. We therefore expect that users will need to revise some aspects of their preambles as the class moves forward. However, this does not affect the *content*, which is the bulk of most presentation sources. As such, we strongly encourage users to transition to ltx-talk.

### Tags for paragraphs

Most of the time a *paragraph* contains only some sentences, without any further structure. However, it is also possible that it has a more complicated structure, e.g., it may contain a quote, or a list, with some text before or after, all belonging to the same paragraph. For example:

```
Don Knuth sends a warm welcome from afar
and writes:
\begin{quote}
  \sffamily Happy TeXing in Calgary
\end{quote}
This is what we did!
```

To model this in the PDF structure, LaTeX surrounds the whole paragraph with a `<semantic-para>` tag; then,

[1] Joseph is rather accidentally the maintainer of beamer!

inside this, each individual text block has its own tag: i.e., in the above example, the text `Don ... writes:` and the text `This is ...!` each have a `<text-block>` tag and the quote has a `<quote>` tag,[2] so you end up with

```
<semantic-para>
 <text-block>
   Don Knuth sends a warm welcome from afar
   and writes:
 </text-block>
 <quote>
  <text-block>
     Happy TeXing in Calgary
  </text-block>
 </quote>
 <text-block>
   This is what we did!
 <text-block>
<semantic-para>
```

We initially called these paragraph-related tags `<text-unit>` (now `<semantic-para>`) and `<text>` (now `<text-block>`), but the original names were difficult to understand, so we changed them.

We also introduced the `<text-fragment>` tag for use inside a `<text-block>` when there is a need for a sub-structure.

### *Provide an `order` key mechanism for templates*

Templates often have to typeset several items of textual data (from keys and/or from template arguments) with slight variations from instance to instance: e.g., the ordering and the separation between the items might differ, or in other cases only a subset of the items are present (or supported). For example, a theorem-like environment typically wants to have a fixed title, a number, some punctuation, and possibly a note (provided through document input). However, one design might ask for "Lemma 3.2 (note)", the next for "3.2 Lemma. (note)", and another for "3.2 Lemma *note*.", with different ordering and changes in the punctuation.

We have now provided code to support this in a general way: e.g., templates can offer an "order" key in which the designer specifies the data items and their separation. Tagging support is automatically provided. The documentation is currently available via `texdoc latex-lab-template`. In due course this will be moved to `lttemplate` in the kernel.

[2] HTML does not have the context of a semantic paragraph and so would mark each of the texts, before and after the quote, with a `<P>`, thus effectively claiming that they are individual paragraphs whilst in reality they together make a single paragraph.

## *Tagged PDF: theorem-like environments*

### *Use the `order` key mechanism for theorem-like environments and proofs*

The theorem-like templates have been rewritten to make use of the new `order` key mechanism. As a result a number of keys got replaced with new (more standardized) ones. Thus, for `title`, `number`, `note`, and `punct` there are now keys for customization with the names `title-decls`, `number-decls`, etc., as well as `title-format`, `number-format`, etc., and these keys all expect an argument.

The default for the `order` key in theorem-like environments is

```
order={title,separator-1,number,
       separator-2,note,punct}
```

To put the number first, or the punctuation directly after the number, simply switch around the names in the comma-separated list.

Note that two distinct separator items are supported (both defaulting to a space), which means that one can easily alter one or both spaces individually simply by setting the keys `separator-1` and/or `separator-2` for a given instance.

The default for `order` in proofs is slightly simpler because proofs aren't numbered; so the template has no `number`, `number-decls` or `number-format` keys and consequently there can be at most a single separator:

```
order={title,separator,note,punct}
```

This also means proofs can now have a note (which is an extension to amsthm) that could be given as

```
\begin{proof}[note={my note}]
```

The default setting of the `note-format` key outputs the note surrounded by parentheses.

### *Fix a color leakage in theorem-like environments*

If the `body-decls` key was given a color specification, then that color leaked back into the caption of that theorem: this is because placing the caption was delayed until a paragraph got started, by which time the body declarations were already in force.[3] This was fixed as part of the `order` key rewrite.

*(tagging-project issues 1225 and 1350)*

### *Subtle improvement for theorem-like captions*

If a theorem caption uses a mixture of italic and upright glyphs, e.g., "*Remark* 2.", then an italic correction should be added to ensure that text doesn't get too close together. In the original implementation that simply worked due to the automated mechanisms

[3] Color in TEX is different from, say, font attributes: if you have already placed some text in a box it still reacts to the current color when the box is placed into the document. In contrast, font changes would have no effect whatsoever at that point.

provided with \textit or \textup but the way the caption is now constructed, the support for tagging (e.g., \pdffakespace) intervenes and the commands can no longer automatically detect whether an italic correction needs adding. We have therefore explicitly added a \/ where necessary. *(tagging-project issue 1632)*

*Theorem-like environment instances for AMS classes*
The AMS classes do not load the amsthm package but inline its code (with the help of docstrip) which makes it complicated to provide first aid code for tagging. The classes also have their own defaults for environments defined with \newtheorem. These layouts are now replicated when the classes are used together with \DocumentMetadata to produce accessible documents. *(tagging-project issue 1632)*

## Tagged PDF: block and list environments

*Revisiting the para-handling logic for block environments*
In LaTeX, block environments like itemize or quote typically scan for a following empty line and if there is none, then the text following the environment is treated as being part of the current semantic paragraph, continuing it: therefore no paragraph indentation is added, nor anything else that signals the start of a new semantic paragraph.

In some situations, or for some types of block environments (such as theorem-like environments), this behavior of continuing the current semantic paragraph is not wanted. We have therefore added a new boolean template key standalone which, when set to true, causes the environment to surround itself with implicit \par commands, mimicking the effect of explicit empty lines in the source.

We took the opportunity to also fix a couple of subtle problems with the mechanism determining the end of the semantic paragraph so that it hopefully works better in a few unusual situations.
*(tagging-project issues 1402 and 1415 among others)*

*Replaced legacy-code key in blockenv template*
The blockenv template had a legacy-code key, used to support some legacy setup for the generic list environment and for the verbatim extension offered by the ltugboat class (where you can write, e.g., \begin{verbatim}[\small] to get a verbatim display in a smaller font size).

However, if the verbatim was directly preceded by text, then the content of the key was evaluated while still in horizontal mode. In that case the \small changed not only the verbatim display but, as a side effect, also altered the \baselineskip of the preceding text. This has been corrected by providing two keys: early-legacy-code, which is evaluated near the start of the template, while possibly still in horizontal mode (used by list); and late-legacy-code, which is evaluated after returning to vertical mode (used by verbatim). *(tagging-project issue 1503)*

*\newlist failed for description lists*
The emulation of enumitem's \newlist declaration failed when the source list was description. This has now been corrected. *(tagging-project issue 1518)*

*Make \item[] produce an empty item label*
After switching to key/value support in \item a side effect was that an empty optional argument was understood as an empty key/value list and not as a request for replacing the generated item label with an empty one. This has now been changed for better compatibility with existing usage. *(tagging-project issue 1557)*

*Extending the enumitem emulation*
The package enumitem is not compatible with the new list code, and in the last release we provided an emulation which supports most of the keys and the \newlist and \setlist commands. This emulation has now been extended. The key labelindent has been added, and the keys leftmargin, labelsep, labelwidth, labelindent and itemindent now support the special value "!". The value "*" no longer gives a low-level error, but for now falls back (with a warning) to the "!" value. Initial support for the ref key has been added too.

## Tagged PDF: heading commands

*New tag names for heading structures*
Until now, heading structures used a mix of PDF tags (Sect), Span and tags from the LaTeX namespace (section, section-number). This has now been consolidated and the LaTeX namespace is used for all heading structure tags.

In the LaTeX namespace we now denote by chapter the complete chapter text (i.e., what was previously the Sect) and use chapter-heading to denote the heading title for the chapter (i.e., the mandatory argument of \chapter). All other document divisions are handled in the same manner, e.g., the namespace now has section and section-heading, etc., right down to subparagraph and subparagraph-heading.

*Use the order key mechanism for heading commands*
In LaTeX release 2026-06-01 we provided a reimplementation of heading commands using templates. The initial set of templates had only limited flexibility, modeling layouts closely related to those from the standard LaTeX classes. As a consequence, they used keys such as number-title-sep to define the space between heading

number and title, on the assumption that both are placed exactly in that order.

We have now written new extended templates to allow for reordering of heading elements and more flexible spacing between them, but in that scenario the original keys no longer make sense and they have therefore been replaced with more generic keys such as `separator-a`, `separator-b`, etc., which do not take a length value but accept general code.

As part of this change we also replaced the `prefix` key (for the fixed text such as "Chapter") by a key named `text` (short for *fixed text*) since this text is not always used as a prefix, depending on the design.

If you have defined instances using the first template version then you can (for now) continue to use them simply by changing the template names in your instance declarations from ⟨*name*⟩ to ⟨*name*⟩`-v1`: for example, you have to change from this

```
\DeclareInstance
    {heading}{chapter}{display} {...}
\DeclareInstance{headformat}
    {heading}{chapter}{display} {...}

\DeclareInstance{heading}
    {section}{display} { ... }
\DeclareInstance{headformat}
    {section}{hang} { ... }
...
```

to this

```
\DeclareInstance
    {heading}{chapter}{display-v1} {...}
\DeclareInstance{headformat}
    {heading}{chapter}{display-v1} {...}

\DeclareInstance{heading}
    {section}{display-v1} { ... }
\DeclareInstance{headformat}
    {section}{hang-v1} { ... }
...
```

No other changes are necessary for now. However, going forward you should switch to the extended templates because the `-v1` templates will eventually be dropped. How to upgrade to the new templates is explained in `texdoc latex-lab-sec-template`.

*Support `force-unnumbered` key in heading templates*

When designing heading commands there are occasions where you want to enforce that a heading always remains unnumbered, even though it should be listed in the table of contents, in bookmarks or in the running header. For this, the heading templates now offer the boolean `force-unnumbered`. It defaults to `false` in which case numbering is determined in the usual way, e.g., though `secnumdepth`, etc. *(tagging-project issue 1473)*

*Support `placement=ragged` in heading templates*

The `placement` key in heading templates was augmented to support the value `ragged` in addition to `page`, `top`, and `normal`. This key value works like `normal` (i.e., the heading can appear anywhere on the page) but if the heading happens to trigger a page break then the previous page will be set "raggedbottom" and not spread out to fill the page in the book or report classes. This is useful on pages that do not have a lot of stretchability in the first place and thus the space between paragraphs is used for stretching which often looks ugly.

It can either be specified on the instance level by setting the `placement` key to `ragged` or for individual headings in the document by setting it in the optional argument to the heading. *(github issue 1844)*

*Heading templates now support column spanning*

In two-column documents top-level headings usually start on a new page and span both columns. In LaTeX $2_\varepsilon$ this was hardwired and headings such as `\chapter` showed this behavior. In the new template-based implementation this is now more flexible and one sets this independently from forcing a new page by setting the key `column-spanning` to `true` (default for `\chapter`) or `false` (default for other headings). Thus

```
\section[column-spanning=true]{title}
```

generates a one-off `\section` heading that spans both columns. Due to the way the LaTeX output routine is currently implemented that also automatically starts a new page, i.e., one can't generate spanning headings in the middle of a page (that can only be done with the multicol package at the moment). That restriction may be lifted when the output routine is rewritten. *(tagging-project issue 1436)*

*Setting left and right margins for (some) headings*

To provide more flexibility in the design of headings the `headformat` templates `display` and `hang` now support the keys `left-margin` and `right-margin`. With their help you can indent or outdent the whole heading on either or both sides. This is in addition to the key `heading-indent` which is a horizontal space at the beginning of the first and often only line.

The `headformat` template `runin` only offers the latter key, as in a run-in heading the heading is on the same line as following text, so the concept of margins for the heading does not apply.

*Logic of `after-penalty-vspace` key slightly improved*

If a page break happens before a heading, then the heading is placed at the very top of the next page without any space (more exactly `\topskip` is applied for the first line of text, but as this is normally equal to 10pt and headings are typically larger the glyphs in the

heading title normally touch the top of the text area of the page).

This behavior changes if the key `after-penalty-vspace` is set to any value different from its sentinel default (`\maxdimen`). In that case the baseline of the first line of the heading will be a `\baselineskip` plus the value of this key down from the top of the page. This means that with `after-penalty-vspace=0pt` the baseline of the heading text will be a full `\baselineskip` away from the top of text area and its glyphs will no longer touch the top.

The most common use for this key is in headings like `\chapter` that always start a new page and want the heading text to drop down by a noticeable amount. But setting it in other circumstances can be useful too: e.g., use `0pt` to ensure that the heading is always placed in the same position, regardless of the glyphs in its title. *(tagging-project issue 1623)*

### *Conditional heading template instances*

In some document designs, different layouts for headings are used depending on the page they are placed on. For example, the heading text always extends into the outer margin, i.e., to the right on recto (odd) pages and to the left on verso pages. The design of headings without a number is also fairly often quite different from that of headings with a number.

Capturing such variations in every `heading` template is possible but fairly complicated and inefficient. For that reason we offer a different interface (similar to that for document "contexts" [1, p. 121, 127]). If the `heading` template instance for the heading command is ⟨*name*⟩ then (any or all of these may be left undefined)

⟨***name***⟩`:recto` is the instance name for recto (odd) pages;

⟨***name***⟩`:verso` is the instance name for verso pages;

⟨***name***⟩`:`⟨***recto-or-verso-or-empty***⟩`:unnumbered` is the instance name for unnumbered headings on recto, verso, or all pages.

If some of these instances are defined then the most specific one matching the current context is used, e.g., a `\section*` command on an odd page will try to use `section:recto:unnumbered` or `section::unnumbered` or `section:recto` or `section` in that order.

Instead of, or in addition to, setting up special `heading` instances it is also possible to use `\IfVersoHeadingTF` inside instance keys. It executes the true or false branch depending on the page the heading is typeset. `T` and `F` variants are also available.

### *Replaced `start-code` and `final-code` keys in heading templates*

The `placement` key in heading templates is responsible for defining the pagination behavior of a heading: e.g., `top` makes the heading start on a new page, `normal` allows the heading anywhere on a page but requires at least two lines of text afterwards, etc.

Internally, this key resets two other keys, `start-code` and `final-code`, to carry out the necessary actions. These keys could also be explicitly set if the available options for `placement` needed further adjustments (in which case `placement` was ignored). Thus, these keys are strongly tied to `placement` (but this wasn't obvious from their names): e.g., you couldn't just set `final-code` without breaking the layout of the heading in surprising ways.

To make this dependency clearer, we have changed the key names to `placement-start-code` and `placement-end-code`, and also updated the documentation of all three keys. The old key names will remain usable for a while, but any code using them should be updated, as these names will eventually get removed. *(tagging-project issue 1617)*

### *Run-in headings followed by displayed math*

When a run-in heading, such as `\paragraph`, was immediately followed by a displayed math environment the heading was inserted too late which confused the tagging support, as the tagging for the formula had already started. This has now been corrected. *(tagging-project issue 1571)*

### *Heading template instances for AMS classes*

In making documents using the AMS document classes accessible we have now reimplemented their headings using the new heading templates (that support tagging). Further work is necessary to make these document classes fully accessible, but the current heading support is a big step towards this. *(tagging-project issue 1477)*

### *Emulation of titlesec package started*

Until now, loading the titlesec package in a document that started with `\DocumentMetadata` (with or without tagging enabled) was not possible: the combination would spit out a number of warnings and then errors because it didn't understand the new template-based heading commands.

We have therefore started to provide an emulation of titlesec that will eventually implement all of its interfaces and can then be automatically loaded as a replacement package.

At the moment all it implements is the package options, e.g., you can use

`\usepackage[sf,sl,`⟨*more options*⟩`]{titlesec}`

and get slanted sans serif headings, etc. However, other important customization commands, such as `\titleformat`, are still missing and will be added in the 2027 June release. Nevertheless, we already now replace

titlesec by this emulation when \DocumentMetadata is used because this improves the situation (no errors and some aspects already work) and can be used for testing.

The progress of this work is tracked in the file latex-lab-titlesec.pdf.

*Tagging of headings in minipages and parboxes*

By default the title and the body of chapters, sections and so forth are surrounded by a <Sect> structure. For this, a heading command first ends previous <Sect> structures of the same and higher level and then starts a new one. If a heading command is used inside a minipage or parbox these <Sect> structures overlap with the box structures and so break the structure tree.

We therefore now simplify the tagging of headings inside such boxes: they tag only the heading title and do not try to produce <Sect> structures. To end a previous <Sect> structure manually before such a minipage, e.g., \tagpdfsetup{sec/end=section} can be used.

## Tagged PDF: other fixes and improvements

*Package fncychap is now tagging compatible*

The fncychap package defines some half-a-dozen chapter layouts that can be selected through options to the package. These layouts can then be further customized through a series of commands defined by the package. Due to its age, and the fact that it overwrote various internals, the resulting chapter headings were not usable when producing a tagged and accessible document.

We have now provided a simple template with appropriate tagging support (but without any customization keys) that makes use of the code in the package and, in this way, supports all the heading designs offered by the package, as well as the customization possibilities it provides. With these fairly minor adjustments the package is now compatible with tagging and can be used when making accessible documents.

Because of the customization possibilities that allow for arbitrary rewrites, we had to make the tagging support fairly general to cover such variants. There is a downside to this: in several designs we end up with a number of empty heading-fragment structures. While this is allowed and doesn't do any harm (other than resulting in some unnecessary processing by the PDF reader) it is not as elegant as a solution with standard templates would be. *(tagging-project issue 570)*

*Package listings is now (mostly) tagging compatible*

The listings package offers various verbatim-like environments and commands for pretty-printing source code. We have now created a style that is loaded automatically after the listings package to make it compatible with tagging.

There are still some open points, e.g., frames are not perfectly tagged, when escaping to math the math is not tagged; when using braces to delimit the argument of \lstinline there is no tagging, but for most standard cases it is now possible to use the package. *(tagging-project issues 70 and 1074)*

*Package float is now tagging compatible*

The existing support for the float package has been extended and the commands \newfloat, \restylefloat and \floatstyle can now be used too. *(tagging-project issue 890)*

*Storing more titles for named references*

When using the nameref package it is possible to reference the names of theorems (as given by the optional argument) and to reference the optional argument of items in description lists. nameref made this possible by patching the internal commands of theorems and lists. These patches no longer worked with the new theorem and list code when \DocumentMetadata is used.

This has now been repaired: Theorems and items with optional arguments will update \@currentlabelname and so allow \label to store this value for later reference. Items will also store the optional argument in \@currentlabel and so \ref will reference this value. For example, given the following input

```
\begin{enumerate}
\item                    First item
\item[1b] \label{item} Second item
\end{enumerate}
See \ref{item} ...
```

then the \ref in last line will produce "1b". *(tagging-project issue 1538)*

*Fixed issue with end hooks for math environments*

Due to an issue in the definition of \math_register_env:nn, adding to the env/⟨*env*⟩/end hook for a math environment such as equation produced an error. The definition of \math_register_env:nn has been fixed and made equivalent to what was previously called \math_register_halign_env:nn. The latter command has been removed. *(tagging-project issue 1603)*

*More control over the tagging of title and headings*

HTML pages and accessibility APIs typically do not distinguish titles as distinct from section headings. Titles are marked up as <h1>, and following heading commands are then marked up as <h2> etc. This allows the title to be announced by screen readers and take part in navigation.

For PDF 1.7, the PDF Association recommends to tag a title like a normal paragraph and to start the tagging of the heading commands with <h1>. Such titles are not announced and do not take part of the navigation.

PDF 2.0 introduced a new **Title** structure element which should in principle allow titles to be distinguished;

however, at the present time it has no defined mapping to the System Accessibility API, and the "Deriving PDF to HTML 1.2" document[4] maps it to `<div>`. As such, tagging with **Title** is equivalent to tagging with **Div** or **NonStruct** and does not produce accessible titles. Titles are in this case simply read as ordinary text.

Given that the standard mappings to the Accessibility API and HTML may change, it is not clear what is the best tagging to use at the present time. The code therefore introduces some options to allow document authors some control. These options can be set with the tagging setup key `title` which can take the values `Title` (the default), `H1`, `html` and `aria`. The default settings maps the title text and part headings to **Title** and the tagging of the other heading commands then starts with `H1`. The option `H1` maps the title text and part headings to **H1** and the other heading commands then start with **H2**. This currently gives the best accessibility experience. A full description of the key and its values can be found in `latex-lab-title.pdf`.

## New or improved commands

### Testing whether a certain template is defined

To better support the declaration of template instances we now provide the conditional `\IfTemplateExistsTF` (and its `T` and `F` variants) so that it is possible to take different actions depending on whether or not a certain template is defined.

### New Lua helper `luatexbase.new_luacommand`

Binding a Lua function to a control sequence with LuaTeX normally requires three steps: allocate a Lua function slot with `luatexbase.new_luafunction`, store the function in the Lua functions table at that index, and then call `token.set_lua` to bind the control sequence to the slot. The new helper `luatexbase.new_luacommand(`⟨*csname*⟩`,` ⟨*fn*⟩`,` ⟨*...*⟩`)` combines these three steps into one call; any additional arguments are passed on unchanged to `token.set_lua`.

*(github issue 2191)*

### Lua callbacks get unique identifiers instead of descriptions

When registering a callback in LuaTeX with `luatexbase.add_to_callback` LaTeX traditionally expected a `description` parameter. In many places this description is used to uniquely identify the callback function which requires the description to be unique. Since this wasn't clear from the name, the parameter has been renamed `identifier`. With this change reuse of the same value for multiple functions is now printing a warning. This is expected to become an error in the near future. *(github issue 1620)*

[4] https://pdfa.org/deriving-pdf-to-html-version-1-2/

### Conditional adding of punctuation

The AMS classes already had a mechanism to conditionally add punctuation characters, e.g., at the end of a heading title if that title didn't already end in a punctuation character. We have now adopted that idea for general use by offering `\AddPunct` (normally for use in template instances) so that one can write, for example

```
title-format = #1\AddPunct{.}
```

As a result, the heading title has a period at its end unless it already ends in a punctuation, e.g., a "!".

*(tagging-project issue 1477)*

### Providing `\strongemph`

In addition to `\emph` to indicate emphasis there is a second concept in PDF and HTML called "strong emphasis", typically signaled using bold face. Some support for this was available in fontspec but not as part of the LaTeX kernel. We have now implemented `\strongemph` directly in the kernel so that it can be used in any document and in particular can generate appropriate tags when accessible documents are produced.

To allow for adjusting its layout we also provide `\DeclareStrongEmphSequence`, `\strongemreset` and `\strongemforce`, acting similarly to `\DeclareEmphSequence`, `\emreset`, and `\emforce` for the `\emph` command.

The default declarations are

```
\DeclareRobustCommand\strongemreset
                     {\bfseries\upshape\ulcshape}
\DeclareStrongEmphSequence{\bfseries,\itshape}
```

which makes `\strongemph` alternate between bold and bold italic (if the font family supports the latter). For font families that offer different grades of boldness a possibly better setup would be to alternate between bold, extra bold, ultra bold (or whatever the font provides). This could be easily adjusted by changing the above declarations.

The mechanism tries to be "smart" by verifying that the given declarations actually alter the current font. If not, it continues and tries the next level or the reset values—the assumption being that there was already a manual font change in the document to the font that is now supposed to be used for "strong emphasis". Of course, this works only if the declarations in the list's entries change the font and not, for example, just the color. In such a scenario one has to add `\strongemforce` to the entry, which directs the mechanism to use the entry, even if the font attributes appear to be unchanged.

*(tagging-project issue 1620)*

### Supporting alternative template key names

To allow template keys to be accessed using multiple names, we have added the command

`\DeclareTemplateInterfaceEquivalents`. This allows key names to be mapped to other key names, either as simple aliases or with more complex relationships. This will enable support for meta-keys (a single richer key which sets multiple simple keys) or for renaming keys as part of development, or providing interfaces that work with slightly different input formats, for example

```
\DeclareTemplateInterfaceEquivalents
  {⟨type⟩} {⟨template⟩}
  { key-e : alias = key-a ,
    key-f : meta  = { key-b = {#1},
                      key-c = {#1} },
    key-g : meta  = { key-d = \UseName{#1} } }
```

To support deprecation, the interface allows specifying a date in an optional argument. If given, then using a deprecated key generates a warning (before the date) to alert users to the need for updating their sources or an error if the date is already in the past, e.g.:

```
\DeclareTemplateInterfaceEquivalents
  {⟨type⟩} {⟨template⟩} [⟨date⟩]
  { old-key : alias base-key , ... }
```

*Modifying template key values*

When making design changes to templates, it is sometimes the case that you want to modify the existing value of a key, rather than set it from scratch. For example, you might have a template where the change required is "make the margins 1 cm narrower" without knowing the current margin value. We have extended the keyval syntax for templates to support this using a `+=` syntax

```
margin-width += 1cm
```

This approach is supported for all key types where it makes sense, for example `length` or `tokenlist`, but not for those that cannot have the value modified, such as `boolean` and `function`.

## Code improvements

*Support for `\UseName` in template key bindings*

We have extended the logic in the key-binding argument for `\DeclareTemplateCode` so that, when using templates, the keyword `name` can be used (in addition to `global`) without any performance impact. This allows automatic variable creation with "awkward" characters, most notably `-` and numerals. Of course, such variables then also need to be referred to via `\UseName` or some equivalent method in the code, so this is useful only if there is a pressing need for such special names.

*Further removal of "fake math" from the kernel*

In the previous release, we removed from the kernel most uses of "fake math" (any use of math-mode that does not denote a formula, but is used only for positioning or manipulating text) [1]. The use of `\underline` in text mode was missed in that update: this has now been addressed.

*`\OmitIndent` now usable in vertical mode*

The command was originally intended only for use in the `para/begin` hook. Now, it additionally issues `\noindent` if encountered in vertical mode.

*(tagging-project issue 1674)*

## Bug fixes

*Avoid problems with page breaks in the middle of verbatim-like environments*

The fix we announced in LaTeX News 41 [1, p. 116] was incomplete: we missed resetting the `\catcode` of comma. This is now corrected. *(github issue 2056)*

*Support nested requests for class or package target dates*

If one loads a package with

```
\usepackage{⟨package⟩}[=⟨date⟩]
```

then the package version current at ⟨*date*⟩ should be loaded. For document classes, `\documentclass` offers the same interface. Inside a package or a class a developer can write conditional code that depends on the requested target date (if any) by using the command `\IfTargetDateBefore`; see [2] for an article on this functionality.

Unfortunately, using the optional argument on a given package also affected the behavior of `\IfTargetDateBefore` in all the following code, i.e., it was not confined to that package code for which this request was made. This has now been corrected.

*(github issue 828)*

## First aids

*Fix amsbook running headers*

The AMS document classes all use the old mark mechanism, but this is no longer provided by LaTeX and therefore these classes need updates. As these are still outstanding we have now made an attempt to provide first aid for amsbook so that running headers hopefully work again. *(github issue 2040)*

*First aid for subcaption used with the corrected amsmath*

In the June 2025 release, amsmath corrected a typo that had been there for some 30 years, so that it now also resets `\label` in `\@arrayparboxrestore`. As a side effect, the `sub@` labels that the subcaption package writes for `\subref` were no longer written to the `.aux` file, and every `\subref` printed "??" (with only a warning). Because the caption bundle is currently not actively maintained, the latex-firstaid package now restores the expected behavior when both packages are loaded. In the long run the proper fix belongs in subcaption itself.

*(github issue 2172)*

## Changes to packages in the amsmath category

### Drop special handling for LuaTeX in \genfrac

The \genfrac command relied on engine functionality to place delimiters around fractions in traditional TeX engines but emulated this functionality in XeTeX and LuaTeX. For LuaTeX this emulation was broken; but it is not needed and it interferes with tagging, so it has been removed. *(github issue 432)*

## Changes to packages in the tools category

### trace: support use with tagging

The trace package adds (to every math formula) code that suppresses unnecessary debugging output; but unfortunately it did this in a way that conflicted with the tagging code for formulas. This has now been corrected. *(tagging-project issue 1633)*

### bm support for (lua-)unicode-math

The bm package has been extended to support some uses with OpenType Math fonts. This will also offer some improvements with classic TeX fonts, adding support for "implicit characters" (command names \let to math characters).

As with classic TeX fonts, \bm does require that the bold math version is set up. For unicode-math this typically requires declaring a bold font using declarations such as the following setting up TeX Gyre Pagella.

```
\usepackage{unicode-math}
\setmathfont{TeX Gyre Pagella Math}
\setmathfont{TeX Gyre Pagella Math}
            [version=bold,FakeBold=2]
\usepackage{bm}
```

The lua-unicode-math package is distributed with packages corresponding to most of the common OpenType math fonts. These packages typically set up the bold version by default. So the corresponding declaration for TeX Gyre Pagella is

```
\usepackage{lum-pagella}
\usepackage{bm}
```

*(github issues 748, 974, 1991, 2180)*

### afterpage could lose pages in some contexts

If a page break occurred near an over-large page, afterpage could fail to reconstruct the document and lose content. The output routine has been adjusted to detect this case, and re-insert the page before inserting the after-page material. *(github issue 1698)*

### showkeys: avoid double printing

Labels could be double printed if the amsmath option leqno was used with showkeys. This should now be fixed. Also, labels for eqnarray were sometimes not shown by this package. *(github issue 118)*

### longtable: separate data for unnumbered tables

Since version 4.21 an empty \LTcaptype produces a longtable without a caption counter. In that case the internal counter for longtables was not stepped either, so such a table shared its link target and the column widths stored in the .aux file with the previous table. This could give duplicate destinations with hyperref, wrong column widths in a long table, and a "Table widths have changed" warning that never went away. The counter is now stepped for every longtable. *(github issue 2227)*

## Documentation changes

### Tagging LaTeX News

We have adjusted the sources for *LaTeX News* (including this news!) to enable tagging (PDF/UA-2) automatically. This applies to the combined newsletters in ltnews.pdf and each individual file ltnews⟨*number*⟩.pdf. This is part of our effort to make all LaTeX documentation fully accessible over time.

### US spelling

We have (finally) decided that standardizing the spelling in the LaTeX sources to use US English is the realistic position.[5] This means that the documentation better aligns with the code. Idioms may still be more varied: this is harder to adjust than spelling.

## References


[1] LaTeX Project Team. *LaTeX 2ε news 1–44*. November, 2026. https://latex-project.org/news/latex2e-news/ltnews.pdf

[2] Frank Mittelbach. *A rollback concept for packages and classes*. *TUGboat* 39:2, 2018. https://latex-project.org/publications/2018-FMi-TUB-tb122mitt-version-rollback.pdf

[3] TeX Users Group. *TUG 2026 talks, Calgary*. *YouTube* 2026. https://youtube.com/playlist?list=PLF0YCVuRyY-o

[4] TeX Users Group. *TUG 2026 Proceedings*. *TUGboat* 47:2, 2026. Available to members and freely available to everyone in spring 2027. https://tug.org/TUGboat/tb47-2/

[5] Ulrike Fischer. *News from the LaTeX Tagged Project: 2026*. *TUGboat* 47:2, 2026. https://latex-project.org/publications/2026-UFi-TUG-tb146fischer-tagging26.pdf

[6] Frank Mittelbach. *Towards accessible document classes*. *TUGboat* 47:2, 2026. https://latex-project.org/publications/2026-FMi-TUG-tb146mittelbach-accessibleclasses.pdf

[7] Timothy Prescott. *Converting a PDF textbook to be accessible*. *TUGboat* 47:2, 2026. https://tug.org/TUGboat/tb47-2/tb146prescott-book-accessibility.pdf


[5] None of the native English speakers on the team are from the US! But the command names of LaTeX are in US English so …