LaTeX News

Issue 43, June 2026 (LaTeX release 2026-06-01)

Contents

Introduction

Progress in embedding tagging code as part of a more general overhaul continues. The implementation of block structures, such as theorems, and headings (sections), has been updated. We are well on the way to providing a flexible interface that many document authors will be able to use without needing additional code or packages.

This work is also an opportunity to look back at (long-standing) LaTeX code and to see where there are issues. As well as refining some spacing, we have addressed a long-standing technical wrinkle: (mis)using math mode simply to align material vertically. This "fake math" is a serious tagging issue but also shows up where the user wants, for example, to apply a color change to all mathematics. With this release, the use of math mode is essentially restricted to "true math".

Working on templates has prompted us to further revise some concepts that have been added to the kernel in recent years, to align the code more closely with the most natural pathways for users.

Work of course is not restricted to tagging. There are further improvements to copy–paste of symbols, several bug fixes for corner cases in amsmath, and more language support for varioref. At the lower level, work on the L3 programming layer continues; notably here, we have made adjustments to better support the upTeX engine in creating programmatic strings.

News from the Tagged PDF project

Setting the language in `\DocumentMetadata`

Until now `\DocumentMetadata` knew only the `lang` key to set the main language. The key was only used to set the language in the PDF catalog and the XMP metadata and didn't affect the language of the document as specified with babel or polyglossia.

With this release we are now adding two more keys: `language` and `other-languages`. The `language` key takes as argument a language in BCP 47 format, so, e.g., `de-AT` or `fr` or `gsw-u-sd-chzh` (Swiss German as used in the canton of Zurich). The value is stored in the document properties and can be retrieved (expandably) with `\GetDocumentProperty{document/language}`.

The value of the `language` key is also used to set the language in the PDF metadata, so it is also a replacement for the `lang` key. If a different (shorter or more generic) value is wanted in the PDF metadata, an optional argument can be used: `language=[de]{gsw-u-sd-chzh}` or the `lang` key can be used after the `language` key to overwrite the value: `language=gsw-u-sd-chzh,lang=de`.

Just like `lang`, the `language` key does *not* automatically change the language setup of a document: it doesn't change the hyphenation patterns or the fixed names and doesn't set class options. It also doesn't force the loading of a language package. The expectation is that in the future, packages that need to know the document language, like babel or polyglossia, will read the value and react accordingly.

The `other-languages` key can be used to declare further languages that are used in addition to the "main" language in the document. The argument is a comma-separated list of languages in BCP 47 format: `other-languages={fr,en-US,ar}`. The list is stored in the document properties too and can be retrieved with `\GetDocumentProperty{document/other-languages}`.

Not setting the language in `\DocumentMetadata`

It is good practice to always set the main document language in `\DocumentMetadata` explicitly using the `lang` or the new `language` key: The value is used to set the `/Lang` key in the PDF and it can then also be used by packages and classes to adapt locale settings.

However, if there is no such setting then the code will use the main language as set by babel or polyglossia. If they are not used then `lang=en` is applied as this has always been LaTeX's default. When we introduced the `lang` key we added a warning to the log in such cases but feedback we received indicated that this caused concerns so now the fallback is applied silently.

(tagging-project issue 1115)

Using `\DocumentMetadata` more than once

Until now it was possible to use `\DocumentMetadata` more than once, the second use only executing key assignments. This was changed; similar to `\documentclass`, the `\DocumentMetadata` declaration will now cause an error if used more than once.

General revision of the block environments

We have now reviewed the implementation of the block environments and among other things consolidated the key names which were initially a bit arbitrary. The block templates are extensively documented in `blocks-doc.pdf` and the full code in `blocks-code.pdf`.

Detect if keys are not applicable to block environments

All block environments accept an optional argument in which the user can specify key settings to overwrite the values normally used for the current instance. Trying to apply an unknown key was caught in case of list environments, but with other environments, e.g., `quote`, such keys got silently ignored. This has now been corrected. *(tagging-project issue 1317)*

Improve handling of consecutive display blocks

If display block environments directly follow each other (without an empty line between them), then they conceptually belong to the same semantic paragraph. This means they should be handled in the same way as if they had directly followed some horizontal text (and not add `\partopsep`, for example). This was already incorrect in LaTeX 2_ε, e.g.

```
Start of paragraph ...
 \begin{center} text-1 \end{center}
 \begin{center} text-2 \end{center}
more text finishing the paragraph.
```

would result in `\partopsep` for the second but not the first `center` environment resulting in uneven spacing. While this is a contrived example, the problem is real and shows up in documents; for documents using `\DocumentMetadata` it has been corrected.

(tagging-project issue 1328)

Making the design of the `description` environment more flexible

In LaTeX 2_ε the layout of the `description` environment was always identical on all nesting levels. In the new implementation, based on block templates, this has been made configurable for each nesting level by providing individual instances for all levels (`description-`⟨*level*⟩). By default, they are all identical.

Emulating the enumitem package

We are in the process of providing an emulation of the enumitem package using the new block templates. Most keys of enumitem are already supported. We have also added support for `\newlist` and `\setlist`.

Other aspects of the package interfaces will follow over time. This is work in progress and the current state is documented in `latex-lab-enumitem.pdf`.

Supporting `\newtheoremstyle`

The AMS classes and the amsthm package offered `\newtheoremstyle` to declare new layout styles for theorem-like environments. The new block implementation now provides an emulation of this interface—at the moment still with a few restrictions.

Some issues with the first implementation of `\newtheoremstyle` have already been corrected; remaining limitations will be addressed in later releases. *(github issues 2092,2093)*

Reimplementing heading commands with templates

In this release we provide a first implementation of heading commands using the template mechanism. This includes a reimplementation of `\@startsection` to support legacy classes and also a (somewhat) limited reimplementation of `\secdef`. This will help in making legacy document classes accessible with minimal code adjustments. Over the summer we intend to add emulations for the titlesec package interfaces and perhaps also reimplementations of other packages using this mechanism, so stay tuned. See `latex-lab-sec-template.pdf` for some extensive documentation.

Improving the tagging of floats

The tagging support code for floats has been overhauled. It now allows tagging support to be added to new float types like listings or tcolorboxes. By default float structures are deferred to the end of the document but it is now possible to switch this on and off and to output the floats in other places in the structure. More details can be found in `latex-lab-float.pdf`.

Enhancing context handling in typesetting

In the 2025 November release of LaTeX [2, p.121] we introduced a prototype implementation for handling typesetting contexts, e.g., to set up different layout for lists in the main galley and in footnotes, etc. After some experimentation we have extended this concept and now support a primary context (as before), a secondary context to sub-structure the primary context (for example, primary context `float` and secondary context current float type, e.g., `table`), and a tertiary (font size) context. For documentation see `latex-lab-context.pdf`. Going forward this is going to be used when we implement caption handling using the template mechanism.

New or improved commands

Recovering instance values

In some cases, editing template instances requires knowing the existing instance values. To support this, we have added the expandable command `\InstanceValue`⟨*type*⟩⟨*instance*⟩⟨*key*⟩, which returns the value if available; otherwise it returns empty if the key or instance does not exist.

Declaring alias counters

Theorem-like environments like lemmas and definitions often use the same counter for the numbering. This makes it difficult to create suitable prefixes when referencing. We have therefore added a new command `\newcounteralias` that allows to create an alias of a counter and so to use a counter under another name. The code is based on a similar command from the package aliascnt. The command `\newtheorem` in LaTeX and also in amsthm has been changed to use `\newcounteralias`. This allows packages like hyperref, zref-clever and cleveref to correctly identify the environment. The following will now reference, as desired, the lemma as "lemma 1" and not as "theorem 1":

```
\documentclass{article}
\newtheorem{theorem}{Theorem}
\newtheorem{lemma}[theorem]{Lemma}
\usepackage{zref-clever}
\begin{document}
\begin{lemma} \label{lem} ... \end{lemma}
In \zcref{lem} we claim \ldots
\end{document}
```

Debugging support for templates

We have added debugging support for template use: this can be activated using `\DebugTemplatesOn`. Unlike template *creation*, the use of templates and instances is intended to be a fast process: as a result, debugging messages use a low-level approach to ensure they can be skipped quickly in day to day use.

Optional argument for the `picture` *environment*

The tagging code extends the `picture` environment to take an optional argument which can be used to add, e.g., an alternative description. This optional argument has now been added to the kernel version of the environment to make it easier for authors to write code compatible with tagged and untagged documents. If `\DocumentMetadata` is not used the optional argument is silently ignored. *(tagging-project issue 1172)*

Commands to store and restore the last skip

The package hyperref surrounds anchors for links with two commands that save and restore the last skip to prevent these anchors from disturbing spacing commands like `\addvspace`. We now provide these commands

under the name `\SaveLastSkip` and `\RestoreLastSkip` directly and use them also in the kernel definition of `\MakeLinkTarget`. *(github issue 2070)*

Code improvements

Revision of handling of "no value" concept

The commands `\NewDocumentCommand`, etc., introduce the idea of differentiating an absent optional argument from one which is empty. When an argument is entirely absent, it is given a special "no value" marker. In previous releases, this was a deliberately-awkward set of character tokens, which are therefore hard to input accidentally.

While this allowed us to easily detect "no value", it turns out there are places we want to be able to add such a value. This comes up particularly in creating templates for some parts of the document structure as part of the wider tagging project.

We have therefore changed the approach to use a marker token, `\NoValue`, and updated the `\IfNoValue(TF)` test and relatives. This means you now *can* type in an optional argument that is interpreted as "not present", but this is not likely to happen by accident.

Revision of keyval conversion for empty arguments

A second revision to `\NewDocumentCommand`, etc., in this release applies to conversion of classical arguments to key–value form, for example

```
\DeclareDocumentCommand \caption
  {s ={short-title} +O{#3} +m}
```

This will convert a classical optional argument of `\caption` to a keyval argument called `short-title`. In this release, if the optional argument is given but blank, the result will be an entirely empty argument rather than `short-title␣=␣`. This reflects the fact that in moving to a keyval model, the meaning of an empty argument is best thought of as an empty keyval list. For cases where a classical `[]` means something is explicitly empty, adding an empty brace group (`[{}]`) will work with both the new code and with older formats.

Revision of `\protected` status of functions in templates

As we use templates more widely, minor adjustments to the workflow are necessary. As part of this, we have adjusted templates such that keys declared as functions are stored with `\protected` status.

Removal of "fake math" from the kernel

Internally, some text constructs in LaTeX have used math mode, as this allowed use of the built-in positioning of material with limited macro effort. Historically, this was important for obtaining useful performance, but today it is more problematic. Tagging requires that such "fake math" is filtered out, and this becomes even more problematic when working with right-to-left languages. (The interaction between text and math directions in LuaTeX is complex.)

In this release, we have removed the internal use of "fake math" from

- `\textsuperscript` and `\textsubscript`
- `\parbox` and `minipage`
- Around `tabular` structures

To avoid as far as possible changes to existing output, the standard settings continue to use math font dimension values for placing super and subscripts.[1] For new documents, we suggest setting the offset to a predictable offset based on text font dimensions (for example a fixed percentage of the x-height).

Developers using systems which check `\showoutput` results, or similar, will see changes both in removal of `\mathon`/`\mathoff` pairs and in some box structures.

Formalization of L3PL (expl3) release requirement

In this release, we have formalized which release of the L3 programming layer (expl3) is required by the kernel: at least the 2023-11-27 release. This has allowed us to tidy up some places where the kernel uses L3PL, in particular where we have made improvements in the L3PL and want these to be used by the kernel. If the version of the L3PL available is too old, an error will be issued and format building will be abandoned.

New or improved documentation

Adding new features to LaTeX means writing new documentation. To date, each new source has had separate documentation, for example `lthooks-doc.pdf`. This is a convenient way for development to take place, particularly prior to code being stable. However, it makes it challenging to find information.

We have therefore started to collect all of this information into a single document, `cmdguide.pdf`. The aim over time will be to include all of the new features here. We also hope eventually to collect the existing (relevant) documentation from all of LaTeX 2_ε , so that this document can then serve as a programmer's reference manual for the kernel, similar to `interface3` for the LaTeX programming layer.

This is work in progress so we expect the document to grow and its structure (based on feedback and experience with its use) might change over time. In particular, the current content is made up of files that were written to be read independently; as such, there is some duplication, suboptimal ordering and formatting

[1] The calculation used for placement of math mode sub- and superscripts is described in Appendix G of *The TeXbook*, and relies on multiple math font dimensions.

variation. That will all be addressed over time, as we aim for a single coherent document.

Glyphs, characters & encodings

Improved copy & paste for pdfTeX and LuaTeX documents
In 2021, additional information was added to the PDF file, when compiling with pdfTeX, from the file `glyphtounicode.tex` in order to improve copying from, and searching in, text for, e.g., common ligatures.

We now extend this support and load additionally a file `glyphtounicode-cmex.tex` which improves copying of mathematical symbols.

Starting with version 1.24, LuaTeX supports an extended syntax of `\pdfglyphtounicode` that makes it possible to improve copy & paste of symbol fonts. Therefore, we load and activate the files also with this engine if PDF mode is active. Other engines do not support the extended `\pdfglyphtounicode` and `\pdfgentounicode`. Here we define the commands as no-op commands to allow packages for symbol fonts to use the commands without testing the engines first.
(github issue 2007)

Bug fixes

Improve transparency of `\label`, `\index`, and friends
Commands such as `\label` or `\index` are supposed to be transparent with respect to surrounding spaces, i.e., spaces on both sides should not lead to several spaces in typeset text. This always worked reasonably well if there is only a single command. However, if there are several of them in a row one could end up with a spurious extra space. This has finally been corrected.
(github issue 1910)

Global mappings for math script font families
The command `\DeclareMathScriptfontMapping` added in the previous release declares related font families used for scriptsize and scriptscriptsize mathematics. Previously this mapping was set locally, therefore it could not be used in `.fd` files. This has been corrected to apply globally like other font declarations.
(github issue 1955)

Changes to packages in the amsmath category

Treat `\dots` before `\xrightarrow` correctly
If one writes `$ a \to \dots \to b $` the result is $a \to \cdots \to b$, i.e., `\dots` are treated as binary dots. If you replace `\to` with `\xrightarrow` then the dots suddenly become comma dots and you have to use `\bdots` to get the binary dots back. This has now been corrected for both `\xrightarrow` and `\xleftarrow`.
(github issue 263)

Don't lose a QED symbol with `fleqn`
The `proof` environment of `amsthm` automatically puts a QED symbol at the end of a proof. Sometimes this is not the best place and in that situation the author can direct LaTeX to place the symbol earlier by using `\qedhere` in an appropriate place. However, when the `fleqn` option was in force this didn't always work and the symbol got dropped in some cases. This has now been corrected.
(github issue 783)

Set the right margin of `multline` to zero with `fleqn`
The `multline` environment of `amsmath` uses wider margins when the `fleqn` option is in force, although only the left margin is intended to change. As a result, the `multline` environment is typeset in a box wider than intended. Visually, this is usually unnoticeable, but it can, for example, make the page box wider than `textwidth`. If an element spans the full page box (such as `hline`), it may then appear wider than the surrounding paragraphs. This has now been corrected.
(github issue 2025)

Correct vertical spacing above `multline`
In multi-line display math environments such as `align` and `gather`, the distance between lines is slightly increased (by an amount controlled by `\jot`) to ensure that tall mathematical symbols do not clash.

It was then discovered that the `multline` environment failed to apply this compensation, resulting in `multline` displays appearing slightly further away from the preceding paragraph than other multi-line equations. This has now been corrected.

As a result, more material might fit on a page, potentially changing the pagination. If that happens, an explicit `\pagebreak` will restore the original pagination in old documents.
(github issue 793)

Improved column tracking and spacing in alignment environments
A few subtle inconsistencies regarding alignment environments were recently identified and fixed:

- The `gathered` environment did not correctly isolate its column count when nested inside other alignment structures, which could throw off the formatting of the parent environment.
- If a user omitted the final `\\` line break at the very end of an `aligned` or `alignedat` environment, the final row would bypass the maximum-column checks and occasionally leave an unwanted trailing space.
- The `aligned` environment did not always reset its internal counter between rows, which could intermittently cause small spacing glitches at the right edge of the alignment.

Now, spacing is consistently applied and excessive columns are accurately detected, even on the final row or when deeply nested. *(github issue 1609)*

Changes to packages in the tools category

Adjustment to the glue used by longtable

Recent LaTeX releases produce ignored warnings about infinite glue shrinkage. The glue in longtable has been adjusted to only use a finite shrink component. *(github issue 1907)*

Hooks for array and longtable

We have added a number of hooks in the array package so that extension packages can add code at defined places without the need to overwrite internals of the array implementation. This will improve the maintenance of such packages because they will then not have to update when there are changes to array itself. The first package to make use of this is fcolumn.

The same hooks are also available in longtable but only if array is also loaded; otherwise they are suppressed. In the future, i.e., if `\DocumentMetadata` is used at the start of new documents array is always loaded.

varioref: Several new variants for German

A number of options were added for German language variants, so that one can now select `austrian`, `naustrian`, `german`, `ngerman`, `swissgerman`, or `nswissgerman`. By default, they all lead to the same strings. *(github issue 1952)*

varioref: Updated default strings for Hungarian

The text strings for Hungarian have been corrected. At the same time a second option was added to select them, i.e., it is now possible to use either `magyar` or `hungarian` with identical results. *(github issue 1977)*

varioref: Chinese locale strings

A `chinese` option has been added to varioref, providing Simplified Chinese locale strings for all reference text macros and format macros. The parentheses in the format macros use full-width characters (U+FF08/U+FF09), consistent with the existing Japanese option.

Changes at the L3 programming layer

The L3 programming layer provides functions to convert between different encodings, which is needed for the creation of PDF metadata, for example. We have recently improved support here for the upTeX engine, which uses a mixed input encoding at the engine level. This should allow more Japanese documents to work directly with standard LaTeX tools for PDF metadata creation, etc.

References

[1] Leslie Lamport. *LaTeX: A Document Preparation System: User's Guide and Reference Manual*. Addison-Wesley, Reading, MA, USA, 2nd edition, 1994. ISBN 0-201-52983-1. Reprinted with corrections in 1996.

[2] LaTeX Project Team. *LaTeX 2ε news 1–42*. November, 2025. `https://latex-project.org/news/latex2e-news/ltnews.pdf`