Quo vadis LaTeX(3) Team — A look back and at the upcoming years

Frank Mittelbach and the LaTeX Project Team

Abstract

This is a brief write-up of a talk given by the author at the TUG'20 online conference.

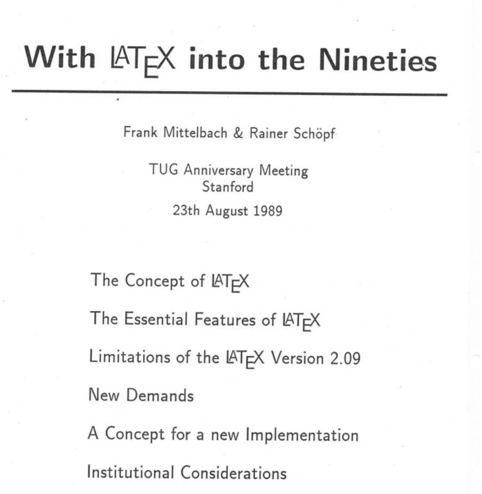
The talk touches briefly on the questions "where we are coming from" (we being the LaTeX Project Team), "where we are now" and then focusses on the LaTeX Project's plans for the upcoming years, which will primarily be focussed on providing an out-of-the-box solution for generating tagged PDF with LaTeX and will include gentle refactoring of parts of the core LaTeX and providing important functionality, such as extended standard support for color, hyperlinks etc., as part of the kernel.

This is a multi-year journey that we have just started and we will briefly explain the places this will take us through. At its end we expect that LaTeX users are able to produce tagged and "accessible" PDF without the need to post-process the result of their LaTeX run.

Contents

1 A quick walk through 30 years of history

In this section we take a short tour from the origins of LaTeX to the present day in order to better understand where we came from and its influence on how we see the future shaping.



With LaTeX into the Nineties

Frank Mittelbach & Rainer Schöpf

TUG Anniversary Meeting
Stanford
23th August 1989

The Concept of LaTeX

The Essential Features of LaTeX

Limitations of the LaTeX Version 2.09

New Demands

A Concept for a new Implementation

Institutional Considerations

Figure 1: The title slide from 1989

The birth of the LaTeX project

A bit more than thirty years ago I gave my first international talk at the 1989 TUG conference at Stanford (Figure 1). There I lectured in front of Leslie Lamport and Don Knuth, boldly pointing out deficiencies of LaTeX 2.09 and what is needed to improve on it.

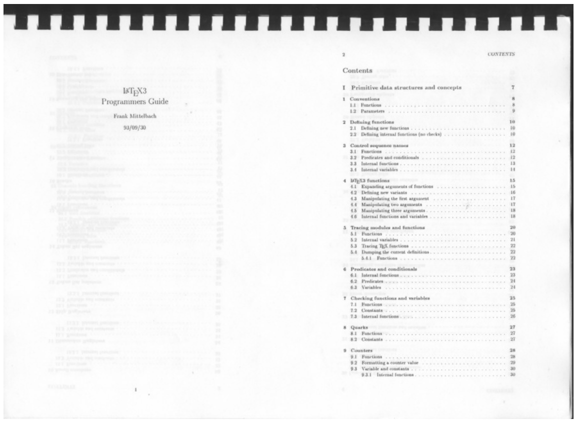
Our criticism wasn't new to Leslie, as we had sent him many bug reports and suggestions during the previous years. And after a long meeting following the talk, Leslie passed maintenance and future development of LaTeX on to Chris Rowley, Rainer Schöpf and myself. For more details on the events back then see the conference proceedings [9].

Leslie continued to work with us, discussing concepts and interfaces, but did not participate in any of the coding for a new version. By the time LaTeX 2ε got released he had fully retired from working on LaTeX (except for sending in the occasional bug report like any other user).

Thus, this day in late August 1989 marked the origin of the LaTeX Project, later often referred to as the LaTeX3 project.

We were young (isn't that always the problem?) and had big plans, but it would certainly be impossible to turn even some of them into reality had we not had the fortune to soon recruit a number of additional members — influential in shaping LaTeX 2ε and beyond.

Few of them will need introductions to anybody who has worked a while with LaTeX, but for the record here are the people beside Chris, Rainer and myself to praise or blame for LaTeX 2ε and many of the packages that you are still using today: David Carlisle, Johannes Braams, Alan Jeffrey, Denys Duchier, Michael



LaTeX3
Programmers Guide
Frank Mittelbach
93/09/30

1

2 CONTENTS

Contents

Figure 2: The 1993 LaTeX3 Programmer's Guide

Downes and Robin Fairbairns. All got their hands dirty in the development of today's LaTeX and/or had a lasting influence on what later became expl3.

The first years

The early nineties were fairly productive years for the team and by around 1992 we had built a complete kernel for a new LaTeX3 system. It was able to compile its own user manual. Figure 2 shows a version from 1993.

In my talk I titled the slide showing this picture "A fully working (useless) LaTeX3 kernel". The reason is that we found out to our dismay that there was serious danger of dying of excessive caffeine consumption while waiting for even a small document to compile (let alone something like that guide).

Basically we were just several computer generations too early (or lousy programmers, or both). We invented, for example, a system comparable to Cascading Style Sheets (CSS) — long before that appeared in browsers. But all these ideas required much more computing power than was available on typical machines back then.

So in the end we gave up and decided that it was a nice but impossible dream.

A new LaTeX version

So instead of continuing with LaTeX3 we cleaned up all the extensions and improvements we had made for LaTeX 2.09, developed a graphics and color abstraction and bundled everything under the name LaTeX 2_ε. This was then promoted as the "newly revised LaTeX standard".

Leslie wrote an updated LaTeX manual [3] and Michel Goossens and myself with the help of Alexander Samarin produced the first LaTeX Companion [2], also known as the Doggie book to many LaTeX users (Figure 3).

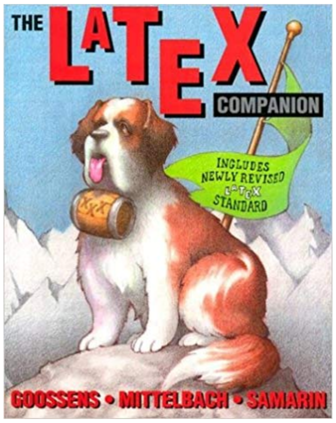

THE LATEX COMPANION
INCLUDES NEWLY REVISED LATEX STANDARD
GOOSSENS • MITTELBACH • SAMARIN

Figure 3: The "doggie" book: the first edition of the LaTeX Companion.

LaTeX 2_ε had a fairly shaky start, largely due to the fact that a small but vocal minority loudly argued against using it and suggested to stay with LaTeX 2.09 instead. The main reason given was that "nobody needs these new 8-bit fonts with precomposed foreign characters and that they take a huge amount of unnecessary space on your hard disk".

However, in the end it took off like a rocket, largely because of all the other goodies it offered — solving many of the problems people were struggling with in the past. If you look through the Web for LaTeX 2_ε you will find a large number of books in various languages that appeared in the following years, clearly showing that there was a high level of interest in the software.

For us it was an important lesson to learn how close success or failure can be, if you have a small but vocal group opposing you.

Highlights of the following decades

Presenting any reasonably complete account of the works of the LaTeX team throughout the years would fill many pages, so here we restrict ourselves to only a few highlights that are relevant for what we intend to do in the future.

Around 1997

At some point we decided to release some of the work we did for LaTeX3 as a package on CTAN named expl3, mainly to preserve it but also because computers had gotten faster, and it seemed that the code could become usable after all at some point in the future.

The new millennium

That action got younger people interested, and first Morten Høgholm and later Will Robertson and Joseph Wright pushed for a complete overhaul of the

Figure 4: The expl3 logo

language. This happened in several phases between 2005 and 2012; see [12, 4] for more details.

Large application packages, such as fontspec, siunitx and others, got written in expl3. At some point the language was evidently in a reasonably stable state and we announced it as fit for general use [13].[1]

Now with a stable expl3 around 2014 we started promoting it and one of the actions was to use a logo for it, which was designed for us by Paulo Cereda — a lovely hummingbird pecking at the "l" (Figure 4).

To indicate that we are moving into new waters I pushed for using the hummingbird also as the official new logo for LaTeX and at some point later we made the switch.

Initially there had been some concerns to make the LaTeX "brand" unrecognizable if it isn't associated with some kind of a lion, but in retrospect it seems fairly clear that the logo was positively received and there is no question these days that this particular bird represents today's LaTeX.

A change in policy

A big step was taken in 2015 when we announced a new bug-fix and enhancement policy. Until then the LaTeX 2_ε format was essentially kept unchanged. Even serious bugs were either not fixed at all, or fixed by adding the fix to the package fixltx2e that one could or could not load as desired.

This meant great stability but it also meant that only the few people who added fixltx2e would benefit from the fixes, while the great majority would stay with the buggy version. In the beginning this was fine but over time it became a burden because packages have to provide alternative code paths based on fixltx2e being loaded or not. We therefore switched to the approach that fixes get applied by default (i.e., everybody receives them) and instead now offer a way (though a rollback mechanism [5, 7]) to opt out, if necessary.

Thus what happened in 2015 was that the accumulated fixes previously in fixltx2e got moved into the LaTeX kernel and the package reduced to an empty shell, unless you used it with an old LaTeX format.

Around that time we also started to bring external developments into core LaTeX. For example, we officially added support for LuaTeX to the kernel and took over the maintenance and development of amsmath from the American Mathematical Society.

Managing future enhancements

But we also went a step beyond bug fixes and integrations. To prepare for future developments we wrote a new testing and distribution environment (l3build [8]) that has been used by us to maintain the kernel sources, and over time also by many other package developers around the world.

A relatively recent activity was to arrange with the major distributions, i.e., TeXLive, MacTeX and MiKTeX, to provide so-called "LaTeX development releases", allowing users and package developers to test pre-releases of LaTeX with ease [6].

We also announced that necessary enhancements to the code (to keep it relevant) would be presented from now on in most cases as opt-out rather than opt-in solutions.

A good example for this policy change is the switch from legacy 8-bit code pages to Unicode, or more precisely to the UTF-8 encoding. This happened in 2018. With the LaTeX release in that year the default input encoding for LaTeX became UTF-8, and in retrospect it is fair to say that few people have noticed any ill effects with their document and had to apply the opt-out. Most people only noticed — if they noticed the change at all — that they could finally use Unicode characters in their documents without problems, a feature that was badly lacking in LaTeX previously.

2 Activities in 2020

Two important changes happened in the spring 2020 release of LaTeX:

- One was a long overdue modernization of LaTeX's font selection scheme to better support all the new high-quality OpenType fonts;
- The other was described in the *LaTeX Newsletter* as "improved load-times for expl3".

Why is the second bullet of any importance? At the time of the release it was indeed nothing more than what it said: users with documents loading expl3 (to begin with, all XeTeX or LuaTeX users) experienced noticeably faster processing times.

[1] The name expl3 stands for "**EX**perimental **P**rogramming **L**anguage (LaTeX) **3**" but it was kept even after it had long ceased being experimental.

But its importance lies in the fact that it marks the end of one era and the beginning of a new one. LaTeX now greets you with

```
LaTeX2e <2020-02-02>
L3 programming layer <2020-06-18>
```

and this means that thirty years after first dreaming about it, LaTeX finally comes equipped with the LaTeX3 programming layer included as part of the format.

3 A look at the future

LaTeX has stayed surprisingly relevant given that its original design dates back to the 1980s.[2] It has, however, limitations, some due to the underlying engine and some due to design decisions made in the past.

Important areas for urgent improvement

Perhaps the most important limitation is that until now LaTeX concerned itself only with producing a "printed result" with paper as the ultimate output medium in mind. Any other usage is either not supported or not directly supported. However, for quite a while now, other usage has become increasingly important. Many documents are never printed or printed only as a secondary action.

LaTeX 2ε added some support for graphics and limited color printing, but otherwise followed the same paradigm. Hyperlinks and other Web publishing support are layered on top, not as integral parts of the design.

As a notable example, the hyperref package has to redefine a larger number of LaTeX's internals and many commands of other packages to be able to achieve its goals and even so is often enough only able to do so by imposing restrictions. Other packages need to patch the same areas, resulting in conflicts and limitations.

Another important issue is that LaTeX very carefully throws away the wealth of structural information it has at its disposal while producing output pages. As a result a PDF or DVI file produced by LaTeX is just a stream of positioned glyphs without much structural information preserved.

If your intention is only to print that document, then this is all that is required, but if you want to produce, say, an accessible PDF document, then a significant amount of structural information and other data has to be embedded into the final output document to guide screen readers, etc., or adhere to the PDF/UA (Accessibility) standard. At the moment this requires extensive manual labor and processes that often have to be repeated after making even minimal changes to the LaTeX source.

Project(s) for the upcoming years

With the challenges outlined in the previous section in front of us we are focussing on a number of areas to address them:

- Embrace and integrate more functionality from existing packages into the LaTeX kernel;
- Provide extended and unified color management, with graphics and font(glyphs) integration;
- Provide standard interfaces for functionality currently available only in an ad hoc way, or not available at all;
- *Enable LaTeX to automatically produce tagged PDF.*

We plan to integrate important functionality from existing packages directly into LaTeX so that it is directly available for user and package writers through standard interfaces. Examples for this are hyperlinks and colors, as already mentioned, but there are several other areas we are looking at.

In addition, we plan to provide standard interfaces for some important capabilities that are currently not available at all or only in rudimentary and ad hoc fashion. An example for this is the hook management system that is planned for the next LaTeX release in fall this year.

Finally, the list contains a one-liner about producing "tagged PDF", which hides a huge project — we will discuss this below.

A focus change — modernize LaTeX through gentle refactoring

When we set out in 1989 to improve LaTeX 2.09 and produce a new version (a.k.a. LaTeX3) the LaTeX universe was largely defined by the software provided by Leslie Lamport and a rather small and manageable number of packages by others. The reason being that it was not at all easy to build applications on top of LaTeX 2.09 and, of course, LaTeX was only a few years in use back then.

When LaTeX 2ε was released in 1994 it solved many of the problems we had initially criticized, even though it wasn't the system we had envisioned — one with a clear separation of user, designer and programmer levels and facilities, which we simply couldn't make work with the existing computing power of those days.

However, LaTeX 2ε offered a package management system with `\usepackage`, command declaration with optional arguments and other goodies for

[2] Or the early 1990s if you think of LaTeX 2ε as the starting point for today's LaTeX.

users and package developers and so over time people started to provide more and more packages for LaTeX that filled the needs of any niche and nowadays several thousand packages for LaTeX are on CTAN.

The increase in the breadth of the software usage over the years made it more and more unlikely that producing a standalone LaTeX3 next to an existing LaTeX 2_ε would gain any traction. It would naturally start out with a very limited scope (because many existing packages would not work with it) and would therefore be unsuitable for most serious usage. But that in turn would mean that as kernel developers we would not get the necessary feedback that ensures that the provided features are meeting the needs of the users and as a package developer there would be no incentive to provide new packages for a new system that isn't widely used — the usual chicken and egg problem.

We have therefore decided that there will be no separate LaTeX3 product in parallel to an existing LaTeX 2_ε. Instead, we will approach the modernization through some gentle refactoring of LaTeX to reach the same target, but in smaller steps.

If you look back at the history outlined earlier, you will see that this journey has already started in 2015 with the new bug-fix policy and the rollback mechanism, which was then followed by the switch to UTF-8 to keep LaTeX relevant.

The strategy we are following here can be outlined in a number of main bullet points:

- Use the L3 programming language to implement all new kernel code now that it is available;
- Replace existing kernel code (over time);
- Keep focus on reliability and compatibility;
- Collaborate with package writers/maintainers to ensure compatibility with kernel changes.

An example of our new strategy is the implementation of a hook management system for LaTeX, which will be introduced in LaTeX in the 2020 fall release.

Hook management as an example

In the past LaTeX offered just a few heavily used hooks, for example, `\AtBeginDocument`. Every other alteration or addition made by a package was done by overwriting existing kernel code, leading to all kinds of known issues.

With the new hook management system, the LaTeX kernel and many packages will get a larger number of hooks in which other packages can add code in a controlled manner, avoiding the need for patching commands. The new system provides standard interfaces for declaring and using hooks, including ways to order code added to hooks by different package in order to resolve package loading problems, and plenty more.

The new system is written in the L3 programming language (the source file is `lthooks.dtx`), but the interfaces are offered in a way that they can be used in all packages, i.e., they do not require the package to be written in expl3 and thus can be retrofitted into updates of legacy packages easily.

The individual hooks provided by the kernel in the first release replace ad hoc solutions in specific areas as provided by packages such as atbegshi, everyshi, atveryend, etoolbox, filehook and others. In future releases, more parts of LaTeX will see hooks added.

Thanks to the *LaTeX development format* concept mentioned above, the new hook management code is already available for testing to anybody interested — which we strongly encourage. As any change to LaTeX will inevitably have ripple effects which need sorting out, such pre-testing is an important part of the overall strategy, to resolve as many problems and borderline cases as possible before new code shows up in the main release.

For the same reason the LaTeX team is actively checking across the huge set of packages supplied in TeX distributions for possible conflicts and working with other developers and maintainers if updates are necessary due to upcoming LaTeX kernel changes. In this particular instance, it was necessary for a handful of packages that patched into existing internal LaTeX commands in places that have been unavoidably changed to support the new hooks.

4 The tagged PDF project

This project is the LaTeX team's answer to the need for preparing LaTeX to uses other than printing on paper. The main goals of this project can be summarized as follows:

- Provide functionality to automatically produce structured PDF, without the need for user intervention or post-processing;
- Provide the necessary interfaces for producing PDF enhanced by features such as "alternative text" (to comply with standards such as PDF/UA).

While the project focusses on PDF as the primary output format, the functionality that needs to be developed will be equally applicable when targeting other output formats that require structured data to be present, e.g., HTML, XML, and new formats such as HINT currently being developed [16, 17].

Background and project status

There has been groundbreaking work done by Ross Moore and others [10, 14, 11] in the last years in the quest for enabling LaTeX to produce “accessible” or more generally “structured and enhanced” PDF.

The unfortunate problem which all these attempts have run into is that it is next to impossible to patch current LaTeX and all needed packages and still obtain reliable and stable results.

A system based on patches is by its nature very fragile, because any change in the patched code will break the system — which will happen regularly if significant patching is needed, as is the case here. In addition, all solutions to date need to enforce severe restrictions on the document content and even then require the user to do serious manual work — largely because of missing machinery and interfaces in LaTeX.

Our plans are therefore to continue learning from this prior work and provide the necessary interfaces directly in LaTeX, so that fragile and incompatible patching is no longer necessary. Some of our initial work in this regard is documented in [15, 1].

What we have undertaken so far with respect to the “Tagged PDF project” is to produce a feasibility study and develop a detailed project plan for reaching the project goals. This is a multi-year undertaking split into six phases and how long it will take will depend in part on the financial backing for the project, i.e., it depends on how much of the work has to be done in our spare time and how much of the development work is financed by sponsors, so that we can have some people work full time on the necessary work.

We are therefore pleased to be able to say that Adobe is sponsoring a fair portion of the estimated project costs, though we hope to attract further industry sponsors and organizations interested in the subject, in order to keep the timeline at a reasonable length.

Project phases and timeline

The project is tentatively divided into six phases progressing in parallel to the LaTeX release cycle; that is, each phase is expected to require one or more LaTeX releases, depending on how much time we can devote to the necessary work.

The deliverables of each phase are expected to be directly applicable to LaTeX users (and developers) so that we can get immediate feedback but also make tangible progress.

Overall, depending on the available financial support, the project timeline is expected to take between three and five years.

Phase I — Prepare the ground

This phase is already well under way and one important deliverable is the introduction of a general hook management system, discussed earlier.

Phase II — Provide tagging of simple documents

The main goal of phase II is to provide automatic tagging of simple documents, excluding more complicated structures such as mathematics and tables. In this phase workarounds are needed for code that will be adjusted later.

This is delivered as a prototype implementation in form of an add-on package.

Phase III — Remove the workarounds needed for tagging

The main goal of phase III is to extend the coverage of automatic tagging and to remove workarounds that were initially necessary to provide a working prototype.

Phase IV — Make basic tagging and hyperlinking available

The main goal of phase IV is to incorporate all the code currently in the prototype packages into the kernel itself. This needs to be done very carefully and cautiously as there should be no negative impact for users processing legacy documents. This is why we expect to need at least a full release cycle for this.

Phase V — Extend the tagging capabilities

With basic tagging available the focus of phase V lies in providing extended support for tagging by adding tables and formulas to the supported elements.

Furthermore, interfaces for specifying alternate text will be developed and added to all relevant elements.

Phase VI — Handle standards

Finally, phase VI will focus on providing additional support for the relevant PDF standards (as far as this is possible using LaTeX directly, without post-processing the resulting PDF), and adding kernel support for outlines and associated files.

Parallel work

In addition to the six phases (which contain tasks that are largely understood from a technical perspective) there are a number of tasks that require research. These will be carried out in parallel to the other work.

Depending on their outcome the structure of the later phases might need some alteration or extension.

5 Stay tuned

Clearly this article provides only a short glimpse of our plans for the immediate and mid-term future. The feasibility study for the tagged PDF project and its implications and dependencies, for example, is a forty page document and touched upon in this document in a few sentences. In the near future we intend to publish this study and more details both on the plans and on our intermediate results.

As a first result from Phase I, you can already now take a look at the new hook management system and provide your feedback for consideration before it get officially introduced in the fall release of LaTeX. With an up-to-date LaTeX installation the relevant commands are:

```
texdoc lthooks-doc   (for documentation)
pdflatex-dev yourfile (for testing)
```

References

[1] U. Fischer. Creating accessible pdfs with LaTeX. *TUGboat* 41(1):26–28, 2020. `https://tug.org/TUGboat/tb41-1/tb127fischer-accessible.pdf`

[2] M. Goossens, F. Mittelbach, A. Samarin. *The LaTeX Companion*. Addison-Wesley, Reading, MA, USA, 1994.

[3] L. Lamport. *LaTeX: A Document Preparation System: User's Guide and Reference Manual*. Addison-Wesley, Reading, MA, USA, second edition, 1994.

[4] LaTeX Project Team. LaTeX3 news, 2009–. `https://latex-project.org/news/latex3-news/`.

[5] LaTeX Project Team. The latexrelease package, 2018. `https://ctan.org/pkg/latexrelease`.

[6] LaTeX Project Team. LaTeX news, issue 30, October 2019. *TUGboat* 40(3):251–254, 2019. `https://tug.org/TUGboat/tb40-3/tb126ltnews30.pdf`

[7] F. Mittelbach. A rollback concept for packages and classes. *TUGboat* 39(2):107–112, 2018. `https://tug.org/TUGboat/tb39-2/tb122mitt-rollback.pdf`

[8] F. Mittelbach, W. Robertson, LaTeX3 team. l3build — A modern Lua test suite for TeX programming. *TUGboat* 35(3):287–293, 2014. `https://tug.org/TUGboat/tb35-3/tb111mitt-l3build.pdf`

[9] F. Mittelbach, R. Schöpf. With LaTeX into the nineties. *TUGboat* 10(4):681–690, Dec. 1989. `https://tug.org/TUGboat/tb10-4/tb26mitt.pdf`

[10] R. Moore. Ongoing efforts to generate "tagged PDF" using pdfTeX. *TUGboat* 30(2):170–175, 2009. `https://tug.org/TUGboat/tb30-2/tb95moore.pdf`

[11] R. Moore. Implementing PDF standards for mathematical publishing. *TUGboat* 39(2):131–135, 2018. `https://tug.org/TUGboat/tb39-2/tb122moore-pdf.pdf`

[12] LaTeX. Project Team. LaTeX news, issue 17. *TUGboat* 28(1):24–25, 2007. `https://tug.org/TUGboat/tb28-1/tb88ltnews.pdf`

[13] LaTeX. Project Team. LaTeX3 news, issue 9. *TUGboat* 35(1):22–26, 2014. `https://tug.org/TUGboat/tb35-1/tb109l3news.pdf`

[14] C. V. Radhakrishnan, Hàn Thế Thành, et al. Generating PDF/X- and PDF/A-compliant PDFs with pdfTeX — `pdfx.sty`. *TUGboat* 36(2):136–142, 2015. `https://tug.org/TUGboat/tb36-2/tb113radhakrishnan.pdf`

[15] C. Rowley, U. Fischer, F. Mittelbach. Accessibility in the LaTeX kernel — experiments in Tagged PDF. *TUGboat* 40(2):157–158, 2019. `https://tug.org/TUGboat/tb40-2/tb125rowley-tagpdf.pdf`

[16] M. Ruckert. The design of the HINT file format. *TUGboat* 40(2):143–146, 2019. `https://tug.org/TUGboat/tb40-2/tb125ruckert-hint.pdf`

[17] M. Ruckert. The HINT project: Status and open questions. *TUGboat* 41(2):208–211, 2020. `https://tug.org/TUGboat/tb41-2/tb128ruckert-hint.pdf`

◇ Frank Mittelbach and
the LaTeX Project Team
Mainz, Germany
frank.mittelbach (at)
latex-project dot org
https://www.latex-project.org