Bumpy road towards a good LaTeX visual editor at Overleaf

Ben Davies

Abstract

Overleaf has both a Code editor and a Visual ("Rich Text") editor. We recently redesigned the Visual editor. Benefits, drawbacks, and specific issues this editor duality poses will be presented, together with some takeaways we have learned on the way.

1 Introduction

Overleaf is a collaborative tool designed to help people work together on a document. We do this by placing one 'true' version in the cloud which everyone works from. However, each collaborator will have different needs, so will need tools to suit them.

For instance, someone working on a document might not know LaTeX or is providing proofreading services and thus can do without seeing the raw form. It is important that we can enable these people to contribute content and make changes to the document without feeling excluded. Further, we want to reduce the burden that can sometimes be placed on those who do know LaTeX in such circumstances.

Another place we hope to make a difference is in learning LaTeX and for those intimidated by code. We want to help lower the barrier to learning LaTeX by making it more familiar and easier to interact with. In these cases, a mode that focuses just on the content but still allows edits to be seen without recompiling would seem to be a useful tool.

2 Road to the Visual Editor

A first attempt at such an editor was our Rich Text mode. This used CodeMirror 5 to decorate parts of the document such as maths (using MathJax) and figures. There were also attempts at 'hiding' code that didn't need to be seen all the time, such as the preamble and common styling commands. Regardless of the decorations, a core principle ensuring that code was always accessible was, and is, maintained.

However, at this time, the Source mode used the Ace editor; as a result, certain features were not available across both modes. Collaborative features such as track changes and commenting were not transferable to the Rich Text mode, limiting collaborators' ability to contribute. This meant people needed to switch back and forth between the modes costing them time and causing distraction.

Given these issues, we decided to migrate both editors to CodeMirror 6. There are several advantages to this outside of the Visual editor, such as better support for accessibility features, mobile devices and non-Latin languages. A key feature is the ability to support both versions of the editor: Code and Visual.

After the migration, we were able to share features across both editors more easily, bringing the collaborative features to the Visual Editor. Not only that, there is now parity between the editors when it comes to, for instance, themes, keybindings and auto-completion. This is because the Visual Editor is now effectively the Code Editor with decorations.

Having completed the migration we are now using an element-by-element design approach to improve the experience. Focusing on specific environments or commands allows us to make more continuous improvements and helps us consider the best behaviours more fully.

3 Features

The best way of experiencing the Visual Editor is to go to Overleaf, start a project and toggle the Visual Editor on. You will find a toolbar containing common actions that also provides familiarity for those new to LaTeX. There's a figure modal so you can insert figures without writing any LaTeX at all.

Given the documents that are usually written in LaTeX it is perhaps unsurprising that the maths environments are also decorated (now including custom maths commands) as well as decorated headings, lists, theorems, text styling and much more.

The Visual Editor was demonstrated at the conference; the video of the talk is available at `youtube.com/c/texusersgroup`.

4 Conclusions

We still have questions about the content we can handle efficiently and how certain things should behave, especially across the two editors. There are also considerations about how much can be done whilst ensuring we retain the full power of LaTeX for all users.

We did learn that providing parity to different code editors (CM6, CM5, Ace) is difficult; we are happier with one. We also learned that just because we know how we would write the LaTeX doesn't mean we know what the 'button' should do and *vice versa*. And finally that providing different interfaces enhances user experience and has been met very positively by the community. So long as we are achieving that, we will keep at it!

◇ Ben Davies
Overleaf
`ben dot davies (at) overleaf dot com`

doi.org/10.47397/tb/44-2/tb137davies-visual