Peter Krautzberger on the web

Written elsewhere

16 Jun 2016

As has been established a number of times, I write better elsewhere. So for the n^{th} time, here are some cross-postings of things I wrote elsewhere.

math formats for the web

Somebody asked on the MathJax user group

> To my understanding MathJax supports these input formats: LaTeX, MathML, and AsciiMath. If I'm making a website and I can choose to use any of the three formats, what are some advantages of choosing each?

Since I've answered this so many times, I thought it might be worth copying here:

“That’s a tricky (trick?) question.

MathML is MathJax’s internal format (essentially anyway) so anything that can be done in MathJax is done through our MathML support, cf http://docs.mathjax.org/en/latest/mathml.html. While MathML is quite good for such an internal purpose, it can be difficult to create. It’s rarely written manually (much like HTML or CSS) and tools can have trouble producing high-quality MathML (converters can fail, editors might produce overcomplicated MathML). MathML is the dominant format used in professional publishing workflows and thus comes with a rich toolchain out of XML-land.

MathJax’s LaTeX-like input provides a faithful implementation of the most common math-mode LaTeX commands as well as other standard packages and a few non-standard features, cf. http://docs.mathjax.org/en/latest/tex.html. LaTeX is much easier to author by hand than MathML and provides the typical LaTeX advantages such as custom macros (for even easier authoring). It also has the benefit of a large community thanks to the wide adoption of TeX as a programming language for print layout in academic writing. LaTeX is probably the most popular format when people have a choice, so MathJax’s TeX-like input has a wide community out there. From a real TeX perspective, MathJax restricts LaTeX input to math-mode since it converts internally into MathML. Due to LaTeX’s print heritage, some constructions are hard to do (e.g., equal-width columns are trivial in MathML but not doable with the default LaTeX macros).

AsciiMath is a lightweight markup language designed to convert well to MathML. I sometimes like comparing it to markdown – not as powerful but much more sensible to write. It does not have the

expressive power of MathML but it is very easy to learn because it was designed by Peter Jipsen specifically for high-school- and college-level students. It is less frequently used but if it's expressive power is sufficient, I tend to recommend it.

In summary, MathML is MathJax's internal format so anything you can do with MathJax you can do with its MathML input. LaTeX is virtually as powerful (with some edge cases), is easier to author by hand, and has a large community both from real TeX-land and MathJax's community. AsciiMath is the little brother of both MathML and LaTeX and provides a good compromise between expressive strength and human readability.

If you look beyond MathJax there are even more options, of course."

Moving on.

math accessibility vs machine readability

On the "Getting Math Onto Web Pages" community group, Tzivya raised a big topic regarding accessibility:

> I would love it the world would come to understand that accessibility is a subset of machine readability. Accessibility APIs are a specialized kind of machine. If we are working on machine readable math, we need to make sure that those specialized machines can read the math too. Otherwise we will do the work twice.

I found myself disagreeing with Tzivya (which means I'm probably wrong because she is **awesome**). This disagreement is mostly

influenced by our work at MathJax for the past year or so, making math rendering accessible via MathJax. But the point is an important one to me because, as I expected (feared?), a few discussion on the Community Group have already brought up the problem of looking for the right™ solution instead of the realistic one.

For me, what we have now is the right solution: HTML, CSS, ARIA, SVG etc, several competing math rendering/computation/etc implementations based on these, lots of tools tangential to them. An excellent kind of mess without standards beyond what works ok for each project out there. It's not the right™ solution but it has the potential of becoming better and better. It's really just another part of web development; nothing else needed.

Anyway, so I wrote:

"I do dream that eventually (maybe 10 years from now?) we'll have a thorough a11y API mapping for mathematics. At the moment, I don't think mathematics (as a culture / language) is ready for this (though web technology probably would be).

Regarding general machine readability vs accessibility, one important difference I see is that machine readability can benefit from partial results whereas accessibility cannot.

A typical example for this might be units. If we can find a way to make units machine readable, I think we'd have a major improvement for STEM on the web. But it won't help accessibility (much) to know that there are units in an expression if it is otherwise unintelligible.

Of course, we currently don't have any standard or best practice for

exposing units on the web. The MathWG had a very old note on units (from 2003) which suggested class='MathML-Unit' on MathML elements; I don't think that's viable approach today. Perhaps schema is a better starting point considering how successful search engines can leverage units in recipes (I could imagine lab protocols and engineering might benefit from similar methods).

For some tools it's extremely easy to generate markup for units, e.g., Jos de Jong's MathJS has a rich interface for handling units and could probably easily expose them in a visual output. TeX has a rich history with the physics and siunitx packages (which are, for example, partially available in MathJax as third party extensions) and heuristics seem feasible to enrich formats in general (again, MathJax can do some of that via the speechruleengine).

I think for humans we have to change our expectations. Otherwise, we'll just end up repeating the mistakes of the past. I'll post some thoughts on the accessibility thread later."

And I then also wrote on the related thread:

"Today the most reliable method is still to use binary images with alt text: static images are the most reliable in terms of cross browser/platform/network conditions for visual rendering and alt-text is the only way to guarantee at least some alternative rendering (e.g. aural and braille) – no matter how poor the results may be.

Don't get me wrong, in many specific situation, there will be better ways. If you have simple content, then you can get decent visual results with HTML tags with nested aria-labels. If you know you

can rely on webfonts (e.g., many ebook situations) then you can use CSS with webfonts for rendering (and again nested labels). If you don't need IE8 (sigh) then you can use SVG etc.

But in generality, binary images with alt-text are still the most robust way – and that's an extremely sorry state. I'm pretty sure we can do better but we need to identify what users need and what tools can realistically achieve today.

My first guess would be: some form of speech text, potentially enabling some level of exploration through nesting (and perhaps full exploration via JS). That's not as bad as it sounds. SVGs with aria-labels are already a close second in terms of usability (pending the ultimate demise of IE8), and like HTML they open up the opportunity of deep-labels and thus already get a certain level of exploration.

But there are other aspects. For example in the US, MathSpeak has a long history and many users of aural rendering are trained to its way of describing the visual structure of an equation. I've heard enough anecdotal evidence to take this very seriously – after all, that's how visual users do it. Still, a few months ago I learned that in Germany, on the other hand, blind students might learn TeX syntax early in school (most likely because there is no tradition like MathSpeak which, after all, precedes the web by decades).

I also expect much overlap with SVG accessibility, where the challenges of summary information at a top level and exploration of details are very similar to mathematics."

Barrierefreiheit

Oh, I gave a talk for Global Accessibility Awareness Day 2016 at the FernUni Hagen – in German (it's been a while). The slides are on

GitHub Pages. It's already somewhat outdated because Wikipedia now serves mainly SVGs (generate with mathjax-node).

Anyway, the core summary stays true:

> Why is it difficult to make formulas accessible? 1. Formulas compress information (extremely) 2. Formulas are often visual 3. Formulas are context-dependent 4. Formulas are poorly authored

In other words, math accessibility sucks bad. And no solution will really help you there. But MathJax now does its best to make it suck less.

Oh, speaking of accessibility, I'm extremely disappointed that I won't make it to role=drinks after all – but if you're close by, why don't you drop by?

To leave a comment write me an email.