MathJax: the Present and the Future

Davide Cervone and Volker Sorge

July 20, 2020

At the Joint Mathematics Meetings in January of 2010, we announced version 1.0 of the MathJax Javascript library for displaying typeset mathematics in web pages. Almost immediately, MathJax became the de-facto standard for including mathematics on the web. It is used in a wide range of on-line journals, including the AMS's own MathSciNet [14] website; it enables on-line blogs, wikis, and question-and-answer sites like StackExchange [17] and Wikipedia [22] to include mathematical expressions; it provides the mathematics for on-line homework systems like WeBWorK [21] and learning management systems like Moodle [15]; and it has been incorporated into e-book readers, screen readers, and similar products [7].

In the ten years since its initial introduction, MathJax has expanded to include TEX and LATEX, MathML, and AsciiMath input formats [8], and several output formats, including HTML-with-CSS, SVG (scalable vector graphics), and MathML [10]. The introduction of sophisticated semantic enrichment and speech-generation functionality in June 2016 [6] has made MathJax a crucial component in generating web pages and e-books that are accessible to readers with visual impairments who use assistive technology like screen readers or Braille output devices. MathJax can make on-line course materials or published research accessible, which is of growing importance in this age of distance learning.

Much has changed in web technology since MathJax was first introduced. New web libraries and improvements to the Javascript language itself have changed the way web-page designers want to use MathJax, and some of the approaches built into MathJax in its early days made it hard to use MathJax in modern web-page workflows. For the past three years, MathJax has been undergoing a complete rewrite from the ground up, with the goal of modernizing MathJaxs internal infrastructure, bringing it more flexibility for use with contemporary web technologies, making it easier to use for preprocessing and server-side support, and making the production of typeset

mathematics faster. The release of MathJax version 3.0 in August 2019 brought these hopes to fruition [5].

In order to make MathJax easier to maintain, version 3 is written in the Typescript language [19], a version of Javascript that allows for adding types to variables and functions that can be checked by a compiler for correctness. This enables errors to be found much earlier, leads to more reliable code that is easier to understand and consequently makes it easier for others to contribute to MathJax. It also allows us to use new features of Javascript that are part of the latest ES6 standard [1], while still supporting older browsers that do not implement them. So, for example we now use ES6's modern class structure [2] and take advantage of asynchronous features like promises [3]. As these features were not available 10 years ago, v2 implemented its own object system and provided custom signals, queues, and callbacks for asynchronous operation.

V3's modern ES6 features, which can be exploited by modern Javascript interpreters for run-time optimization, together with its new internal design removed some performance issues that were inherent in the design of v2, improving the rendering speed of MathJax. Because the two versions operate so differently, it is difficult to make precise comparisons, but in tests that render a complete page with several hundred expressions, we see a reduction in rendering time of between 60 and 80 percent, depending on the browser and operating system.

MathJax v2 also used its own loading mechanism for accessing its components, which did not work well with modern Javascript packaging systems like webpack [20] or rollup [16]. Version 3 resolves that problem, so it can interoperate better with modern web workflows. One can make custom single-file builds of MathJax, or include it as one component of a larger asset file.

One key feature in v3 is the ability to run MathJax synchronously, and in particular, to provide a function that can translate an input string (say a TeX expression) into an output DOM tree (say an SVG image) [12]. This was not easy in version 2, since its operation was inherently asynchronous. With MathJax 3.0, this is straight-forward and can be applied to both individual expressions as well as entire documents. It is particularly important for the preparation of rendered documents for offline consumption.

MathJax was designed originally for use in a web browser, but this left unaddressed the desire to pre-process mathematics on a server. MathJax v3 was designed to make this possible, as it can be used within node applications in essentially the same way as in a browser [**?**, 13]. That is, you can load MathJax components, configure them through the MathJax global variable,

and call the same functions for typesetting and conversion as you do within a browser. This makes parallel development for both the browser and server much easier. Moreover, node applications can access MathJax modules directly (without the packaging needed for MathJax's use in a browser). This gives the most direct access to MathJaxs features, and the most flexibility in controlling MathJaxs actions.

Making mathematics accessible to readers with disabilities has been an important feature that distinguishes MathJax from other math-rendering solutions. While originally aimed to work with third party assistive technology software of the time, since version 2.7 MathJax also offers its own accessibility solution that is meant to work independently of browser, operating system and assistive technology solutions (e.g., screen readers) a user employs. The extension offers for readers with special needs in particular those with visual impairments (e.g., blindness or low vision) or cognitive impairments (e.g., dyslexia or dyscalculia).

The accessibilty extension uses the speech rule engine [18] library for translating mathematical expressions into speech strings. Its core feature comprise the automatic voicing of formulas together with their interactive navigation, synchronised highlighting and an abstraction feature that allows for visual simplification and summary speech description of formulas. These features have not only been ported to MathJax v3, but greatly extended, making use of v3 new modular setup to allow for a flexible pick-and-mix style personalization of accessibility tools.

In particular, MathJax now offers a number of different rule sets for voicing mathematics, that can be pre-selected or switched on the fly. That is, during interactive exploration a reader can choose another rule set, resulting in a different way an expression is spoken, thus providing a different view on the expression. Low vision support has been improved by enabling magnification not only for an entire formula, but also for sub-expressions during interaction. A further emphasis is to provide better accessibility to advanced mathematical material exploiting information gained from the original LaTeX code, to generate more appropriate speech for different areas of Mathematics but also for subjects like Physics, Chemistry and Logic.

Although MathJax originally was intended as a stop-gap measure until browsers implemented native math rendering (through MathML), after more than ten years, browser support for MathML is not universal, and MathJax continues to bring quality math typesetting to all modern browsers. Our work has been supported by grants from the Sloan Foundation and the Simon's foundation, as well as generous contributions from our sponsors, including the AMS, SIAM, Elsevier, IEEE, and a variety of professional

societies, publishers, and web sites [11], without whose ongoing financial support, MathJax would not have been possible. The version 3 rewrite puts MathJax in a strong position to continue to make beautiful and accessible mathematics available on the web for the next ten years.

References

[1] ECMAScript 2020, http://ecma-international.org/ecma-262/.

[2] ES6 classes, https://developer.mozilla.org/Web/JavaScript/Reference/Classes.

[3] ES6 promises, https://developer.mozilla.org/Web/JavaScript/Reference/Global_Objects/Promise.

[4] MathJax home page, https://www.mathjax.org.

[5] MathJax version 3, https://www.mathjax.org/MathJax-v3-0-5-available/.

[6] MathJax accessibility features, https://docs.mathjax.org/en/latest/basic/accessibility.html.

[7] MathJax in use, https://docs.mathjax.org/en/v2.7-latest/misc/mathjax-in-use.html.

[8] MathJax input formats, https://docs.mathjax.org/en/latest.

[9] MathJax node demos, https://github.com/mathjax/MathJax-demos-node#mathjax-demos-node.

[10] MathJax output formats, https://docs.mathjax.org/en/latest/output/index.html.

[11] MathJax Sponsors, https://www.mathjax.org/#sponsors.

[12] MathJax typesetting commands, https://docs.mathjax.org/en/latest/web/typeset.html.

[13] MathJax web demos, https://github.com/mathjax/MathJax-demos-web#mathjax-demos-web.

[14] MathSciNet, https://www.ams.org/mathscinet/.

[15] Moodle, https://moodle.org.

[16] Rollup, https://rollupjs.org/guide/en/.

[17] StackExchange, https://stackexchange.com/.

[18] Speech Rule Engine, https://speechruleengine.org.

[19] Typescript, https://www.typescriptlang.org.

[20] Webpack, https://webpack.js.org.

[21] WeBWorK, https://webwork.maa.org.

[22] Wikipedia, https://www.wikipedia.org.