Nice looking system emails with MJML

Author
Ruud Renssen

Design Development

Publish date
2017-05-09

Did you ever design and develop an email template? Chances are you spent a great deal of time on the compatibility of your email to make sure it renders good in every client.

There are a ton of ways to receive your email: in Outlook '13 on your PC, in Mail on your iPhone, in Gmail's webmail client on your Galaxy Tablet or any other possible combination. It's a jungle of devices and clients, all with their own rules and constraints. This makes creating nice looking system emails a time-consuming effort and a real pain in the back. A shame, because **receiving feedback from an application by mail is often an integral part of the user experience.**

TL;DR: Show me the example on GitHub

MJML tries to solve this problem. You can build your templates within the clean MJML markup language and after it's processed you have an HTML template with all the clients' exceptions and specifics, much like Sass processes to CSS. This makes building an email template a lot easier and less time-consuming.
To test whether or not this solution is feasible, I've built an email library as proof of concept. The library consists of:

- Welcome mail
- Monthly bank statement
- Password reset mail
- Newsletter

Getting started with MJML

Because there are already a dozen articles about installing the MJML framework, this article won't cover the installation process. It's important to know that both the framework and the utilities are npm-packages.

Although MJML has its own syntax, it does not take a lot of time to learn. MJML is structured as an XML document and looks a lot like HTML. MJML offers default elements for common building blocks like buttons, tables, images and so on; these default elements are called components. If you want something really exotic and reusable, you can build your own component.

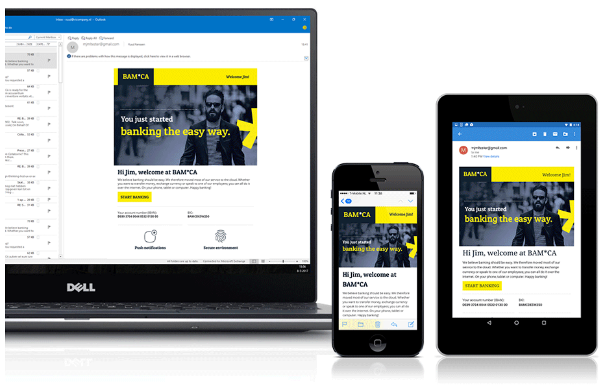
MJML has another advantage: it's (kinda) responsive. By nesting columns, you can create a template that has a single column on (most) small screens and more columns on larger screens. Most times email is read on a mobile device, so this could actually prove to be quite valuable.

Building and testing an email library with MJML and MJML-utils

One of MJML most time-saving features is that you have classes at your disposal AND you can make use of partials/includes. Setting up shared components and classes is therefore really easy. In the example, all the different templates include both elements.mjml and classes.mjml. In elements.mjml all the default elements are defined. In classes.mjml all custom classes are defined and, when included, available within the template. Appearance can now be managed from a central point and is applied to all templates at once. This also makes for much better maintenance because there is no duplicate code.

Unfortunately, MJML is no real template engine, and variables and loops are therefore not available. Although this is a deliberate choice from the makers, it would be handy to pass some content into partials. The header, for example, is always the same except for the content. And a for-loop would be very useful for newsletters.

For testing and building another npm-package comes in handy: MJML-utils. This enables building and watching an entire folder and send out test emails. For sending test emails you need a Gmail account. Combined with a Litmus account, this makes for a very swift workflow.



Conclusion

MJML seems to be a suitable tool for system emails. There are a lot of constraints, but since email has a lot of constraints, maybe it's for the better. I've experienced some render issues with the Outlook client, but was genuinely surprised by how consistent it worked across different devices and clients. I plan to do a Litmus test to see if this assumption is correct.

Pros

- Huge time saver
- Very handy for maintaining multiple emails with different layout
- Testing emails is very easy with mjml utils
- Custom fonts if supported
- Responsive possibilities if supported

Cons

- Dependence on MJML framework
- No variables or loops at your disposal
- Some problems with Outlook: hero images don't work and a white line appears

Don't forget to check MJML's website: mjml.io. All example code is available on GitHub. If you have any questions, don't hesitate to ask: ruud@vicompany.nl.

Previous
Our first dotNed meeting

Next
Something different from your usual line chart