WeasyPrint – Position Paper

WeasyPrint is an HTML+CSS to PDF rendering engine written in Python. Created 9 years ago, it is free and open source software, downloaded about 400,000 times per month. It aims to scrupulously support web standards for printing and to keep simple, clean, easily hackable source code.

The Good

Generating printable documents out of HTML and CSS gets really easier with time. New CSS properties introduced during the last years open exciting layout opportunities, and we especially appreciate them since we know that they require a lot of work from the specification editors.

Features provided by print-only specifications (Paged Media, Generated Content for Paged Media) are really useful, solve most of real-life problems when creating documents and are quite easy to use for people with basic knowledge of CSS and automatic document layout.

Even better: many features largely used in printed documents have also been added or transferred in other specifications. `box-decoration-break` in Fragmentation, `bookmark-*` in Generated Content, `hyphens` in Text are good examples of generic CSS properties needed for printed documents.

As these properties are found in various specifications, they probably are more appealing for major web browsers than print-only properties. Having a lot of different implementations is really important as it helps to find specification issues and reach stability sooner.

The Bad

Even if specifications are continuously getting better, some features are currently missing according to our users needs:

- there should be a way to select the last page of the document and the last page of a page group;
- table headers and footers can be repeated on each page, but there is no CSS property to activate or deactivate this behavior;
- `bleed` could allow 4 different lengths for the 4 page sides.

The Ugly

Our main problem with CSS print is the Working Draft status of paged media specifications.

We have implemented a lot of CSS properties in WeasyPrint, we have an endless amount of work remaining to implement them all, and we know how to live with this frustration. But relying on drafts is more frightening than frustrating as it may require a lot of work for features that may be rejected or changed, possibly years after they have been added.

Changing the implementation also means that page designers will have to change their stylesheets in the future. These modifications give end-users the impression that renderers based on unstable specifications are unstable software. We cannot deny that it is actually true.

For example, the `running()` value is a good source of possible future modifications. This is a must-have feature giving the possibility to include formatted text in page margins.

Below the value definition, another syntax is proposed in an Issue block, with the possibility to move elements rather than just copy them. This syntax is already proposed in the 2014 version of the specification.

Unfortunately, even if this new syntax is appealing, we cannot implement it with confidence. Having to rely for such a long time on Working Draft specifications gives time for users to learn features and include them in their documents. As major page-based features are in draft status, it is also not possible for implementations to wait for the specifications to reach the candidate recommendation status.

Prefixing properties is not a viable solution, as they do not prevent stylesheets to be modified when the property changes. They, at least, warn authors that these features may change in the future, but they also add a small complexity that may be source of bug reports and decrease CSS readability.

The Future

Year 2020 is very important for WeasyPrint. This project has been developed for 9 years by Kozea, a small company, as a side project. It is now widely used, with an increasing number of followers and contributors.

In order to make the engine reach higher quality, speed and innovation goals, a new structure will be created in a few weeks, dedicated to WeasyPrint and its related libraries —covering SVG rendering, hyphenation, CSS parsing, cascade, etc. This organization will be independent, with its own business model, its own agenda and dedicated resources.

This moment is an opportunity for us to invest more time contributing to Web standards.

We would like to help the W3C to solve specification issues faster and to give CSS print the needed momentum. We definitely would like to discuss with other implementations teams and give more feedback to the W3C.

Guillaume Ayoub
WeasyPrint lead developer
February 2020