LaTeX.css Paged Layout

Vincent Dörig

May 2020

Abstract

LaTeX.css is an almost class-less CSS library that turns your HTML document into a website that looks like a LaTeX document. Write semantic HTML, add `<link rel="stylesheet" href="https://latex.now.sh/style.css">` to the `<head>` of your project and you are good to go. The source code can be found on GitHub at https://github.com/vincentdoerig/latex-css.
This is a paged CSS layout for LaTeX.css, using Paged.js. Both together display the document paginated in the screen, and allow to export it to PDF format through the browser.

Contents

Getting Started

- Add `<link rel="stylesheet" href="https://latex.now.sh/style.css">` to the `<head>` of your website or install the package using `npm install latex.css`.
- (optional) Add any classes to elements described in the next section.
- (optional) If you need support for LaTeX math, add the following script to include MathJax:

```
<script id="MathJax-script" async
src="https://cdn.jsdelivr.net/npm/mathjax@3/es5/tex-mml-
chtml.js"></script>
```

- (optional) If you need syntax highlighting, add the following script to include Prism and the Prism LaTeX stylesheet (or use any other):

```
<link rel="stylesheet"
href="https://latex.now.sh/prism/prism.css">

<script src="https://cdn.jsdelivr.net/npm/prismjs/prism.min.js">
</script>
```

- (optional) If you want the document to be displayed in a paginated layout (so you can print it to a PDF file, for example) add the Pajed.js script (see detailed section about Pajed.js loading).
- Done.

Class-based Elements

Author and Abstract

Use the following code to add an author and abstract to your document. It will look like [this](#).

```html
<p class="author">John Doe <br> December 7, 2020</p>


<div class="abstract">
  <h2>Abstract</h2>
  <p>...</p>
</div>
```

Theorems, Definitions, Lemmas and Proofs

Theorems, definitions, lemmas and proofs are supported. Just wrap your content in a div and add the corresponding class to the element like in the following example.

```html
<div class="theorem">...</div>
<div class="definition">...</div>
<div class="lemma">...</div>
<div class="proof">...</div>
```

Below are some examples.

Proofs & Theorems

Theorem 1. *The real numbers $\mathbb{R}$ are uncountable.*

Proof.   If $\mathbb{R}$ is countable, then [0, 1] is countable as well. Hence there exists a map C from $\mathbb{N}$ onto [0, 1] with

$$C(n) = \sum_{i=1}^{\infty} c_i(n) 10^{-i}$$

where $c_i(n) \in \{0, 1, \ldots, 9\}$, are the digits in decimal expansion. Now consider a real number

$$x = \sum_{i=1}^{\infty} \bar{c}_i 10^{-i} \in [0, 1]$$

with $\bar{c}_i \neq c_i(i)$. Obviously $C(n) \neq x$ for all $n \in \mathbb{N}$. Hence C is not onto. A contradiction.   q.e.d.

Theorem 2. *If S is both countable and infinite, then there is a bijection between S and $\boldsymbol{N}$ itself.*[1]

Proof.   For any $s \in S$, we let $f(s)$ denote the value of k such that s is the k the smallest element of S. This map is well defined for any s, because there are only finitely many natural numbers between 1 and s. It is impossible for two different elements of S to both be the k the smallest element of S. Hence f is one-to-one. Also, since S is infinite, f is onto.   q.e.d.

Lemmas

Lemma 3. *An even number plus an even number results in an even number.*

Definitions

Definition 1. A definition is a a statement of the meaning of a word or word group or a sign or symbol.[2]

Paragraphs

In order to get automatic first line indentation of paragraphs, like in default LaTeX articles, the class `indent-pars` can be used with `article` tag.

```
<article class="indent-pars">
  ...
</article>
```

The CSS style provided follows `babel` specific language rules: by default, the first paragraph after a section title is not indented in English, unlike Spanish and French languages.

To avoid first line indentation of some specific paragraph, the class `no-indent` can be used.

```
<p class="no-indent">...</p>
```

---

1. From https://www.math.brown.edu/~res/MFS/handout8.pdf. ⌢

2. "Definition." Merriam-Webster.com Dictionary, Merriam-Webster, https://www.merriam-webster.com/dictionary/definition. Accessed 18 May. 2020. ⌢

Language Support

The labels of theorems, definitions, lemmas and proofs can be changed to other supported language by including the following snippet, linking to the desired language in addition to the main CSS file.

```
<link rel="stylesheet" href="https://latex.now.sh/lang/es.css" />
  ...
<html lang="es">
```

Take a peek at the language support demo to see how the labels of the different languages change.

Sidenotes

Sidenotes can be used as an alternative to footnotes, where the user does not have to jump to the bottom of the page to read it. [1].

[1] Yay, sidenotes!. The sidenote is assigned to the right margin of the text (they will be finally displayed or not, depending on the sidenotes CSS style used).

Sidenotes do need a little bit of setup, they are made up of a label, an invisible checkbox on top of the number and a span with the text inside. The superscript is set automatically and incremented using CSS when the checkbox has a class of `sidenote-number`.

```
<label for="sn-1" class="sidenote-toggle sidenote-number"></label>
<input type="checkbox" id="sn-1" class="sidenote-toggle" />
<span class="sidenote"><!-- sidenote content --></span>
```

If you do not need superscripted numbers, you can opt out of the `sidenote-number` class and the sidenote will not have a number assigned.

This is a sidenote without a number.

The snippet for a sidenote without a number is very similar:

```
<label for="sn-anything" class="sidenote-toggle"></label>
<input type="checkbox" id="sn-anything" class="sidenote-toggle" />
<span class="sidenote"><!-- sidenote content --></span>
```

Add a class of `left` to the span with the `sidenote` class to make the note be assigned on the left side of the page instead of the right. [3]

Alternative Typeface

LATEX.css uses Latin Modern by default. If you don't like this typeface (hard to read, too thin, etc.), you can switch to the <u>`Libertinus`</u> font family by adding a class of `libertinus` to the `<body>` tag.

HTML Elements

For a preview of all HTML elements with LaTeX.css, check out the <u>HTML5 elements test page</u>.

Text Formatting

This sentence is **bold**. If you like semantics, you might go with **strong** or *emphasized* text. If not, *italic* is still around. Small text is for fine print. Your copy can also be $_{\text{subscripted}}$ and $^{\text{superscripted}}$, <u>inserted</u>, ~~deleted~~, or <mark>highlighted</mark>. You would use a <u>hyperlink</u> to go to a new page. Keyboard input elements like `Cmd` + `Shift` are used to display textual user input.

Blockquotes

> Give me six hours to chop down a tree and I will spend the first
> four sharpening the axe. — *Abraham Lincoln*

Definition Lists

Definition Title One

First definition description

Binomial theorem

$$(x + y)^n = \sum_{k=0}^{n} \binom{n}{k} x^{n-k} y^k = \sum_{k=0}^{n} \binom{n}{k} x^k y^{n-k}$$

Tables

---

3. A sidenote on the left margin (but it will be transformed into a footnote, because of the configuration used).

Table 1. A sample table with a descriptive caption.

Header 1	Header 2	Header 3
Description 1	Description 2	Description 3
Description 1	Description 2	Description 3
Description 1	Description 2	Description 3
Footer 1	**Footer 2**	**Footer 3**

Table 2. Example table taken from <u>this paper</u> on how to produce quality tables with LaTeX.

Item		
Animal	**Description**	**Price ($)**
Gnat	per gram	13.65
Gnu	stuffed	92.50
Emu	stuffed	33.33
Armadillo	frozen	8.99

Table 3. Color names and values.

Name	HEX	HSL	RGB
Teal	#008080	hsl(180, 100%, 25%)	rgb(0, 128, 128)
Goldenrod	#daa520	hsl(43, 74%, 49%)	rgb(218, 165, 32)
Cornflowerblue	#6495ed	hsl(219, 79%, 66%)	rgb(100, 149, 237)
Lightcoral	#f08080	hsl(0, 79%, 72%)	rgb(240, 128, 128)

Images

Mountain landscape by [John Towner](#).

Miscellaneous

Syntax Highlighting

If you need syntax highlighting for code, LaTeX.css provides a [PrismJS](#) theme that immitates the [minted](#) package for LaTeX. Add the following stylesheet and script:

```html
<link rel="stylesheet" href="https://latex.now.sh/prism/prism.css">

<script src="https://cdn.jsdelivr.net/npm/prismjs/prism.min.js">
</script>
```

And use it like this:

```html
<pre>
  <code class="language-html">
    <!-- your HTML code snippet -->
  </code>
</pre>
```

Change which languages Prism highlights by customising the script [here](#).

Paged Layout

Paged.js Loading

If you want to paginate the document with Paged.js, you only need to add some links to `latexcss-paged.js` and `paged.polyfill.js` scripts, add a link to the CSS paginated style file, `style-paged.css` (together with `style.css`), and load the HTML file in a web server.

In order to work properly, Paged.js must be the last one of the scripts that processes the document content. `latexcss-paged.js` provides a custom configuration to easily load Paged.js together with MathJax and Prism.js if they are found. Specifically, you must be careful that `latexcss-paged.js` is executed before Paged.js starts, so the later can use the provided configuration. For this reason `async` attribute should not be used with script links. Also, `style-paged.css` should be added after `style.css` and `prism.css` files. You can follow a scheme like this:

```html
<head>
  [...]
  <link rel="stylesheet" href="https://latex.now.sh/style.css" />
  <!--add optionally prism.css if you are going to use it-->
  <!-- <link rel="stylesheet"
href="https://latex.now.sh/prism/prism.css" /> -->
  <link rel="stylesheet" href="https://latex.now.sh/style-paged.css"
/>
</head>
[...]
<!--add optionally MathJax or Prism.js if you need them-->
<!--
<script id="MathJax-script"
src="https://cdn.jsdelivr.net/npm/mathjax@3/es5/tex-mml-chtml.js">
</script>
<script src="https://cdn.jsdelivr.net/npm/prismjs/prism.min.js">
</script>
-->
<script src="https://latex.now.sh/latexcss-paged.js"></script>
```

```
<script src="https://unpkg.com/pagedjs/dist/paged.polyfill.js">
</script>
```

If you need to configure a more complex `PagedConfig` settings (for example, to load some other javascript libraries), `LatexCss` provides a startup promise, `LatexCss.startupPromise`, so you still can use it as follows:

```
<script scr="url/to/someScript.js"></script>
<script>
[...]
  window.PagedConfig = {
    before: function () {
      return Promise.all([someScriptStartupPromise,
window.LatexCss.startupPromise()]);
    },
  };
</script>
<script src="https://latex.now.sh/latexcss-paged.js"></script>
<script src="https://unpkg.com/pagedjs/dist/paged.polyfill.js">
</script>
```

However, you must be very careful with the scripts you load and the "lazy loading" HTML elements you use, or you can easily break the document display.

Fallback loading method. If you still have problems loading other scripts, you can disable automatic Paged.js previewer and call `window.PagedPolyfill.preview()` after setting a sufficient delay time:

```
<script>
  window.PagedConfig = {
    auto: false,
  };
  setTimeout(() => {
    window.PagedPolyfill.preview();
  }, 1500);
</script>
<!-- Add optionally latexcss-paged.js (if you want some of the
features it provides) -->
<!-- <script src="https://latex.now.sh/latexcss-paged.js"></script> --
```

```
>
<script src="https://unpkg.com/pagedjs/dist/paged.polyfill.js">
</script>
```

Finally, if you want to print the document: click "Print..." entry in your browser main menu and then select "Save as PDF", making sure "Margins" and "Headers and Footers" options are set to "none".

Page Size and Margins

By default, an A4 page size is used for paged layout and side notes are not displayed. Because web browser's incomplete CSS Paged Media implementation, some CSS must be wrote if you want change the size of the page or the width of margins. For example, you can use the following CSS code (after `style-paged.css` link)

```
[...]
<link rel="stylesheet" href="https://latex.now.sh/style-paged.css" />
<style>
  @page {
    size: letter portrait;
    margin-top: 100pt;
    margin-left: 100pt;
    margin-bottom: 100pt;
    margin-left: 100pt;
  }
</style>
```

if you want to change the document size to **US letter** and increase the width of the margins.

Utility Classes

Page Numbers in ToC

The class `toc-page-numbers` can be used with `body` tag to display the page numbers corresponding to the different entries of the table of contents.

Sidenotes

Side notes are not displayed by default. To show the side notes, only on one of the margins, the following classes can be used with `body` element:

- `sidenotes-outer`. Show the side notes on the outer margin, hide those on the inner margin (because only single-sided documents are supported at this moment, inner margin is the left margin and outer margin is the right one);
- `sidenotes-inner`. Show the side notes on the inner margin and hide the side notes on the outer margin.

By default, side notes automatically adjust to the size of the margin. However, the width of the side notes can be set manually using the CSS variable `--sidenote-width`, in the same way as for margin width:

```
[...]
<style>
  :root {
    --sidenote-width: 80pt;
  }

  @page {
    margin-top: 100pt;
    margin-right: 80pt;
    margin-bottom: 100pt;
    margin-left: 160pt;
  }
</style>
```

Text Properties

Three classes to adjust the font size of the document are provided: `font-size-10`, `font-size-11` and `font-size-12`, for 10pt, 11pt and 12pt font size respectively.

There are also three classes available to change the line spacing: `line-height-single`, `line-height-onehalf` and `line-height-double`, if you want single, one and half or double line spacing respectively.

Page Break Control

Two CSS classes are provided to *suggest* Paged.js make a page break: `break-before-page`, `break-after-page`. You should keep in mind that Paged.js follows its own rules and sometimes refuses to obey.

There are also other three CSS classes to try to prevent page breaks: `break-before-avoid`, `break-inside-avoid`, `break-after-avoid`.

If you use many CSS page breaks/avoids in a row, Paged.js usually fails to comply with instructions.

Paged.js Tweaking

There are two different points to discuss: how the document is displayed in the screen and how the document is printed, corresponding to `@media screen { ... }` and `@media print { ... }` respectively.

Starting from version 0.4.1 Paged.js removes any CSS rule inside `@media screen { ... }`, both for screen display and print; you must add `pagedjs-ignore` so Paged.js skips this media block, passing it to the screen display:

```css
@media screen, pagedjs-ignore {
    /* CSS rules that will affect only the display on the screen */
}
```

Paged.js includes a `style` element in the header of the html file whith its default CSS styles (see [here](#)). Also, reads and processes all your CSS styles and files and inserts them in the same way.
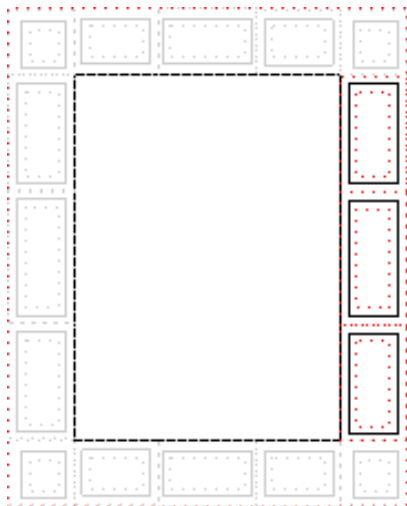
Paged.js design

Paged content design is organized following this scheme:

```
div.pagedjs_pages > div.pagedjs_page > div.pagedjs_sheet >
div.pagedjs_pagebox
```

`div.pagedjs_sheet` is arranged as a grid including sheet bleeds; `div.pagedjs_pagebox` is arranged as a grid, containing page margins. And `div.pagedjs_pagebox` is page box from [CSS page model](#)

The design of `pagedjs_pagebox` consist of four corner margins, and four group of margins distributed on a grid with three columns and rows.

Grid design for `div.pagedjs_pagebox`
(from [W3C CSS Paged Media Module](#) docs)

You can learn more about Paged.js page layout [here](#)

Some Paged.js Variables

You can assign values for some of the following variables, but they also can be automatically populated by Paged.js. For example if you use a CSS style like

```
@media print {
  @page {
    size: A4 portrait;
  ...
  }
...
}
```

the values for `--pagedjs-width` and `--pagedjs-height` (among others) will be populated with the right A4 size paper dimensions. Here is a list of some of Paged.js variables, their names are autodescriptive:

- `--pagedjs-width, --pagedjs-height.`
- `--pagedjs-margin-top, --pagedjs-margin-right, --pagedjs-margin-bottom, --pagedjs-margin-left.`

- `--pagedjs-width-right`, `--pagedjs-height-right`, `--pagedjs-width-left`, `--pagedjs-height-left`.
- `--pagedjs-pagebox-width`, `--pagedjs-pagebox-height`.
- `--pagedjs-page-count`.
- `--pagedjs-page-counter-increment`.
- `--pagedjs-footnotes-height`.
- `--pagedjs-footnotes-count`.

You must also be careful where the variables are populated/updated, in order to get the right values[4].

---

4. This is another footnote, with a more complex structure, using the dagger symbol (†) as footnote mark (you can't see the dagger in the footnote mark because all the footnotes are automatically numbered after being processed by Paged.js).