Documents in HTML, printed to PDF

papyrus is a documents engine. You write a document as HTML or React pages. On screen each page is a sheet of A4 or US Letter paper, and the same pages print to a PDF. A report or a resume stays one source for the web and for print.

It checks every PDF before writing it: the number of pages in the HTML, the PDF's own page count and the count you expect must agree. If they don't, for example because a page overflowed onto an extra sheet, the export stops instead of writing the file.

papyrus is young. Versions are dates (YY.M.R), not semantic versions, so any release may break things. Pin an exact version and read the changelog before you upgrade.

Two packages, one version

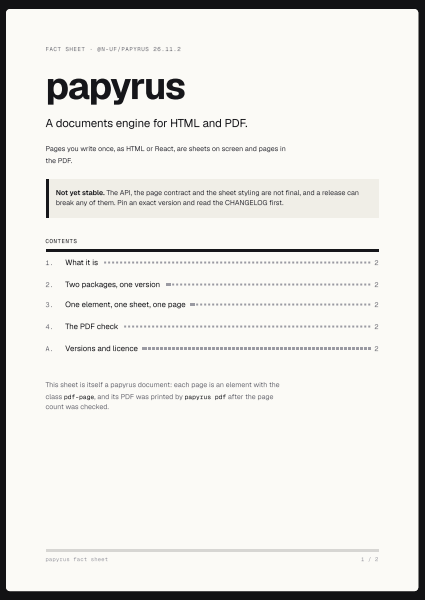
- **@n-uf/papyrus** ↗, the engine. Plain ESM JavaScript with type declarations and no framework, so it runs in a plain HTML page, in a bundled app and in Node. It holds the page contract, the sheet sizes and print CSS, the document registry and the PDF exporter.
- **@n-uf/papyrus-react** ↗, React components on top of the engine: page and content blocks, a document registry typed for React, and print styles. It ships TypeScript source.

Both carry no brand, copy or content of their own; your documents supply those. They are released together at one version, on npmjs with provenance and on GitHub Packages.

DOCUMENTS

Fact sheet

A4 · 2 pages ⬇ PDF 169 KB


FACT SHEET · @N-UF/PAPYRUS 26.11.2

papyrus

A documents engine for HTML and PDF.

Pages you write once, as HTML or React, are sheets on screen and pages in the PDF.

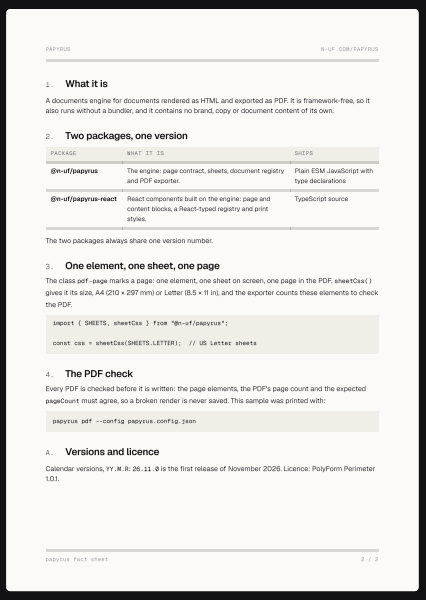
Not yet stable. The API, the page contract and the sheet styling are not final, and a release can break any of them. Pin an exact version and read the CHANGELOG first.

CONTENTS

1. What it is 2
2. Two packages, one version 2
3. One element, one sheet, one page 2
4. The PDF check 2
A. Versions and licence 2

This sheet is itself a papyrus document: each page is an element with the class `pdf-page`, and its PDF was printed by `papyrus pdf` after the page count was checked.

papyrus fact sheet 1 / 2


PAPYRUS N-UF.COM/PAPYRUS

1. What it is

A documents engine for documents rendered as HTML and exported as PDF. It is framework-free, so it also runs without a bundler, and it contains no brand, copy or document content of its own.

2. Two packages, one version

PACKAGE	WHAT IT IS	SHIPS
`@n-uf/papyrus`	The engine: page contract, sheets, document registry and PDF exporter.	Plain ESM JavaScript with type declarations
`@n-uf/papyrus-react`	React components built on the engine: page and content blocks, a React-typed registry and print styles.	TypeScript source

The two packages always share one version number.

3. One element, one sheet, one page

The class `pdf-page` marks a page: one element, one sheet on screen, one page in the PDF. `sheetCss()` gives it its size, A4 (210 × 297 mm) or Letter (8.5 × 11 in), and the exporter counts these elements to check the PDF.

```
import { SHEETS, sheetCss } from "@n-uf/papyrus";

const css = sheetCss(SHEETS.LETTER);  // US Letter sheets
```

4. The PDF check

Every PDF is checked before it is written: the page elements, the PDF's page count and the expected `pageCount` must agree, so a broken render is never saved. This sample was printed with:

```
papyrus pdf --config papyrus.config.json
```

A. Versions and licence

Calendar versions, `YY.M.R`: `26.11.0` is the first release of November 2026. Licence: PolyForm Perimeter 1.0.1.

papyrus fact sheet 2 / 2

Release notes

Letter · 1 page ⬇ PDF


@N-UF/PAPYRUS

Release notes

26.11.0 to 26.11.2

Same code throughout. All three releases ship the same code as 0.4.1; what changed is the versioning, the licence and where the package is released from.

26.11.2

- The package's homepage is now its overview page, n-uf.com/papyrus. Same code as 26.11.1.

26.11.1

- Now released from its own repo, n-uf/papyrus, to npmjs (with provenance) as well as GitHub Packages. Same code as 26.11.0; the package now ships this CHANGELOG.

26.11.0

- Calendar versions (`YY.M.R`) from now on, like hypr-tiling, doodl and debugjs; `@n-uf/papyrus-react` shares the version. Same code as 0.4.1: `^0.4.x` ranges do not pick this up, so move them to `^26.11.0`.
- License: PolyForm Perimeter 1.0.1 (was `UNLICENSED`); see `LICENSE`.

Source: `packages/papyrus/CHANGELOG.md` in n-uf/papyrus.

@n-uf/papyrus release notes 1 / 1

SLIDES

Deck

16:9 · 3 slides ⬇ PDF 113 KB


DECK · @N-UF/PAPYRUS 26.11.2

papyrus

Documents in HTML, printed to PDF

papyrus deck · n-uf.com/papyrus 1 / 3


THE PAGE CONTRACT

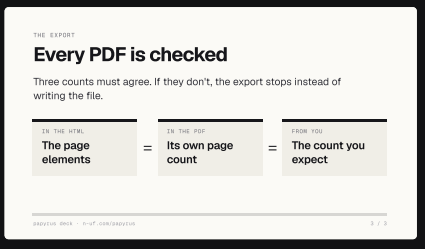
One element, one sheet, one page

Every element with the class `pdf-page` is one sheet on screen and one page in the PDF. `sheetCss()` sizes it to A4 or Letter.

```
import { SHEETS, sheetCss } from "@n-uf/papyrus";

const css = sheetCss(SHEETS.LETTER);  // US Letter sheets
```

papyrus deck · n-uf.com/papyrus 2 / 3


THE EXPORT

Every PDF is checked

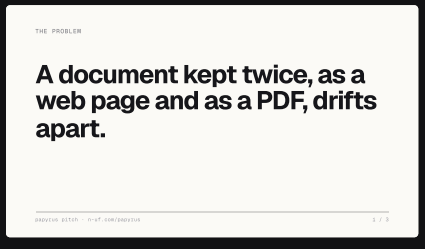
Three counts must agree. If they don't, the export stops instead of writing the file.

IN THE HTML		IN THE PDF		FROM YOU
The page elements	=	Its own page count	=	The count you expect

papyrus deck · n-uf.com/papyrus 3 / 3

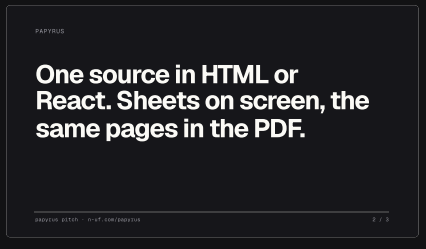
Pitch

16:9 · 3 slides ⬇ PDF 61 KB


THE PROBLEM

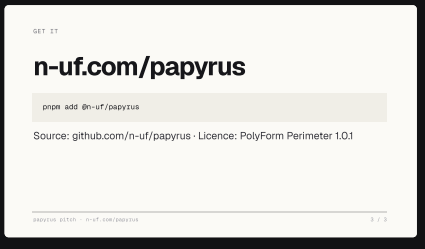
A document kept twice, as a web page and as a PDF, drifts apart.

papyrus pitch · n-uf.com/papyrus 1 / 3


PAPYRUS

One source in HTML or React. Sheets on screen, the same pages in the PDF.

papyrus pitch · n-uf.com/papyrus 2 / 3


GET IT

n-uf.com/papyrus

```
pnpm add @n-uf/papyrus
```

Source: github.com/n-uf/papyrus · Licence: PolyForm Perimeter 1.0.1

papyrus pitch · n-uf.com/papyrus 3 / 3

One element, one sheet, one page

Every element with the class `pdf-page` is one sheet on screen and one page in the PDF. `sheetCss()` sizes it to A4 or Letter and keeps one per printed page; on screen the sheets sit on a grey canvas.

```
import { SHEETS, sheetCss } from "@n-uf/papyrus";

const css = sheetCss(SHEETS.LETTER);                    // US Letter sheets
const alt = sheetCss(SHEETS.A4, { pageClass: "page" });  // your own page class
```

Pages without a bundler can link the ready-made `@n-uf/papyrus/sheet-a4.css` or `sheet-letter.css` instead.

Numbered tables of contents

A document is registered with its title, expected page count and table of contents. `numberToc()` numbers the entries, appendices included.

```
import { docRegistry, numberToc } from "@n-uf/papyrus";

docRegistry.register({ id: "report", meta: { title: "Report", pageCount: 4, toc
}, component: render });
numberToc(toc); // Map: entry id -> "1.", "2.", "A." (appendix) or null (skipped)
```

Export to PDF

The `papyrus` command prints every document in a config file through headless Chromium (Playwright). The server that renders the documents must already be running (`--wait` gives it time to answer):

```
papyrus pdf --config papyrus.config.json
```

For plain HTML files no server is needed: point `baseUrl` at the folder.

```
{
  "baseUrl": "file:///Users/me/resume/src/",
  "docUrl": "{id}.html",
  "outDir": "../dist",
  "docs": [{ "id": "resume", "sheet": "letter", "pageSelector": ".page",
"pageCount": 3 }]
}
```

- **Sheet:** A4 by default, `"letter"` for the whole config or one document. It sets the viewport, the print CSS and the PDF paper.
- **Pages:** `pageSelector` names your page elements when they are not `.pdf-page`; `pageCount` is the number you expect.
- **Fonts:** web fonts are awaited before printing (up to 10 seconds), so pagination doesn't depend on load timing.
- **Stale PDFs:** list a document's source folders and `papyrus pdf --check` reports committed PDFs whose sources changed since they were printed.
- **Browser:** set `PAPYRUS_CHROMIUM_PATH` to print with an installed Chrome instead of Playwright's download.

The same export runs from code with `renderPdf(browser, url, options)` from `@n-uf/papyrus/pdf`. `createPdfHandler` from `@n-uf/papyrus/server` renders a document to a downloadable PDF on request, in a serverless function or any server with a Web `Request`. In development, the Vite plugin `papyrusPdfPlugin` from `@n-uf/papyrus/vite` serves `/docs/<id>.pdf`.

React components

`@n-uf/papyrus-react` gives you `PDFPage` and blocks for text, tables, lists, cards and indicators. `registerDoc` (from `/dossier`) registers a React document, `useDocParam` (from `/params`) keeps a document parameter in the URL, and `printStyles` (from `/styles`) holds the sheet CSS and print fixes. Branding is yours: `Logo` and `DocumentFooter` render only what you pass.

```
<Logo name={<MyWordmark />} variantTextLines={{ home: ["MADE BY"] }} />
<DocumentFooter title="Report" copyright="© 2030 Acme" />
<TOCItem number="1" title="Intro" page="3" section="market" sectionColors={{
market: "rgb(14, 116, 144)" }} />
```

With Tailwind v4, import the package's stylesheet: `@import "@n-uf/papyrus-react/tailwind.css"`.

Install

```
pnpm add @n-uf/papyrus            # engine only, no bundler needed
pnpm add @n-uf/papyrus-react      # React components
```

In a Next.js app, add `@n-uf/papyrus-react` to `transpilePackages`. `playwright` and `vite` are optional peer dependencies, needed only for the PDF export and the Vite plugin.

Source on GitHub: n-uf/papyrus ↗. On npm: @n-uf/papyrus ↗ and @n-uf/papyrus-react ↗. Changelogs: engine ↗, React components ↗. Licence: PolyForm Perimeter 1.0.1 ↗.