# JetsonPDF.Fluent

## Comprehensive Feature Showcase

This document exercises every public extension method, *decorator*, layout container, leaf element, and page slot in **JetsonPDF.Fluent**. Each section is anchored —click any TOC entry below.

## Contents

## 1. Page slots: Header, Footer, Background, Foreground

Every page in this document already demonstrates the four non-content slots: the grey **Background** tint, the title + page-number **Header**, the dynamic-component **Footer**, and the rotated translucent **DRAFT** watermark in the **Foreground**. Slots are drawn z-stack: BG -> Header -> Content -> Footer -> FG.

Back to TOC

## 2. Column

Stacks children vertically with an optional Spacing between items. Auto-paginates at child boundaries.

first item

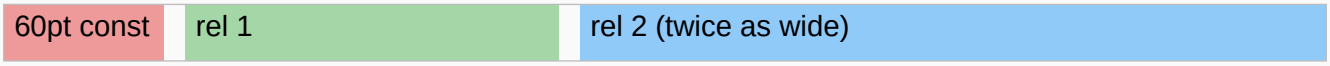
second item

third — taller content

Back to TOC

## 3. Row

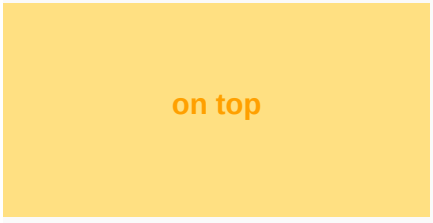
Mixes ConstantItem(width) and RelativeItem(weight). Constants resolve first; relatives split the remainder.




Back to TOC

## 4. Stack (z-stack)

Children draw on top of each other from the slot origin. Reported size is the max of any child's measured size.




Back to TOC

## 5. Inlined (horizontal flow with row-wrap)

Items pack left-to-right at intrinsic width and wrap to the next row when the slot is exhausted.

alpha beta gamma delta epsilon zeta eta theta iota kappa lambda mu nu xi omicron

Back to TOC

## 6. Grid (fixed-column flow)

Items distribute across N equal-width columns then wrap to the next row. Row height = max of items in that row.

| | | | |
|---|---|---|---|
| #0 | #1 | #2 | #3 |
| #4 | #5 | #6 | #7 |

Back to TOC

## 7. Table — header + footer + column-span + row-span

Header rows repeat on every page. Footer renders only on the last page. **Cell(columnSpan: 2)** widens; **Cell(rowSpan: 2)** makes a cell occupy multiple rows; auto-flow honours occupancy.

| Group | Item | Status |
|---|---|---|
| **Alpha** | Implementation | Done |
| | Tests | Done |
| | Docs | WIP |
| **Beta** | Spec review | Open |
| | Prototype | Open |
| end of items | | |
| | **Total items** | **5** |

Back to TOC

## 8. Sizing: Width, Height, Min/Max

Width / Height pin a dimension exactly. MinWidth widens reported size; MaxWidth clamps the slot before measuring.

80×30 MinWidth 120 MaxWidth 160 (rel)

Back to TOC

## 9. Spacing: Padding variants

Padding(all), Padding(h, v), and PaddingLeft/Right/Top/Bottom + PaddingHorizontal/Vertical for axis-only padding.

| .Padding(8) | .Padding(12, 4) | PaddingLeft(20) |
|---|---|---|

Back to TOC

## 10. Alignment: horizontal + vertical fill

Horizontal alignment fills width and chooses x; vertical alignment fills height and chooses y. Combine by chaining.

| L/T | C/M | R/B |
|---|---|---|

Back to TOC

## 11. Visual decorators: Border, Background, AspectRatio

| Border(2) | Background | AspectRatio 3:1 |
|---|---|---|

Back to TOC

## 12. Transforms: Translate, Rotate

Translate offsets at draw time (measurement unaffected). Rotate pivots about slot center via PDF CTM.

| Translate(8, 8) | Rotate(20°) |
|---|---|

Back to TOC

## 13. Pagination control

Decorators that interact with the pagination loop. **ShowOnce** renders on the first page it'd render then becomes Empty; **SkipOnce** is Empty first time then renders normally; **ShowEntire** promotes Partial -> Wrap so an element never splits across pages; **EnsureSpace(min)** wraps to next page when less than min height remains.

This banner is wrapped in .ShowOnce() so it only appears on the FIRST page that renders this section. Re-running the showcase against a smaller slot would suppress it on continuation pages.

Back to TOC

## 14. Text: simple, styled, rich

Plain Text("...") uses the inherited DefaultTextStyle.

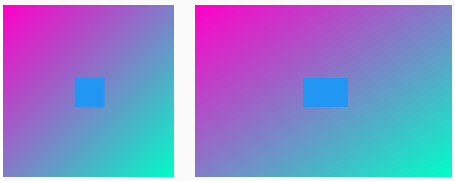
*Styled Text("...", s => ...) overrides the style for this leaf.*

Rich text supports multiple *styled* runs in one paragraph: **bold**, colour, size, and inline page references like Page **5** / **9** . Lines wrap at word boundaries preserving per-run styles.

Back to TOC

## 15. Image

.Image(pdfImage) fills the slot at the natural aspect; .Image(pdfImage, w, h) sets explicit dimensions.



<- A 64x64 RGB PNG generated in-memory. Right: same image stretched to 96x64.

Back to TOC

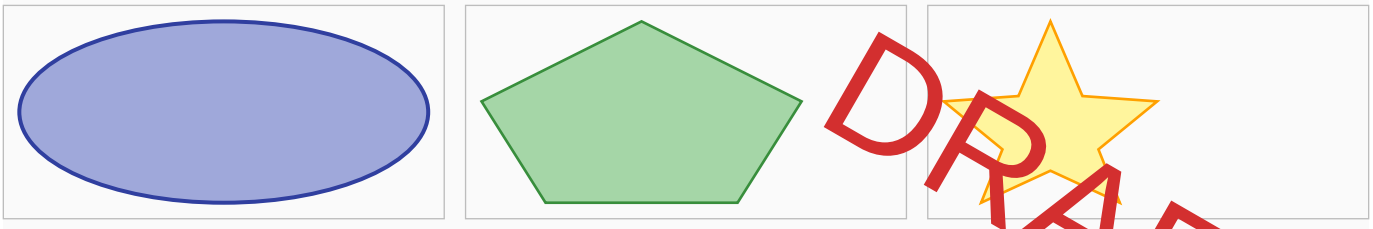
## 16. Lines and Placeholder

LineHorizontal / LineVertical fill their direction; Placeholder is an empty slot for layout scaffolding.

Back to TOC

## 17. Shapes: Ellipse, Polygon

Ellipse fills its slot's bounding box; Polygon takes engine-coord points relative to the slot's top-left.




Back to TOC

## 18. Shadow decorator

.Shadow(dx, dy, color) paints a flat offset rectangle behind the child — the 'card shadow' idiom. Not a Gaussian blur, but a small offset reads as a soft drop shadow at most viewing zooms.

Shadow(2, 2)

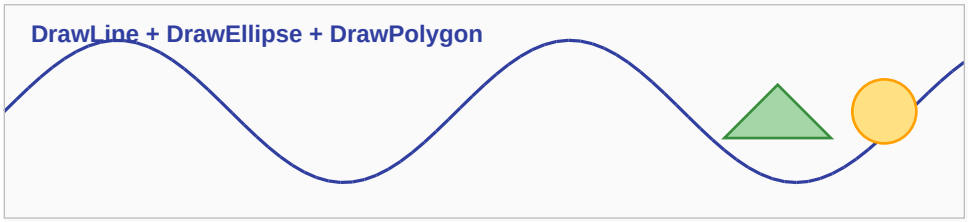
Shadow(4, 4)

Shadow(0, 6)

Back to TOC

## 19. Canvas escape hatch (IDrawingSurface)

Canvas((surface, w, h) => ...) hands the slot to a user callback with engine-coords drawing primitives. Use for anything the layout primitives can't express directly.




Back to TOC

## 20. Page numbers: standalone and inline

Standalone:

PageNumber: 6 TotalPages: 9

Inline rich text: this is page **6** of **9** —both go through deferred-text resolution so digit width is reserved at measure time and the actual digits are stamped after the layout pass.

Back to TOC

## 21. Form widgets

All five AcroForm widget types: text field, check box, combo box, list box, push button. Each is a leaf that consumes the slot.

Text field: default

☒ Check box (initially checked)

Combo box:

List box:

One
Two
Three
Four

Submit

Back to TOC

## 22. Links: external URL and internal navigation

Click this link to open https://jetsonpdf.com —but note rich-text spans aren't link-wrapped; the actual link is below.

Visit jetsonpdf.com (external link)

Jump to the Table section (internal link)

Back to TOC

## 23. Dynamic components

IDynamicComponent.Compose(ctx) is invoked once per page. ctx.CurrentPage is current; ctx.TotalPagesEstimate is the final total (after the renderer's two-pass count).

This text was composed dynamically for page 1 —the running count at composition time. (For 'Page X of Y' headers prefer the rich descriptor's deferred TotalPages.)

Back to TOC

## 24. Colors palette and FromHex

~50 named colours plus FromHex("#RRGGBB") for arbitrary RGB.

| | | | | |
|---|---|---|---|---|
| Red | RedLight | RedDark | Pink | Purple |
| PurpleDark | Indigo | IndigoDark | Blue | BlueLight |
| BlueDark | Cyan | Teal | TealDark | Green |
| GreenLight | GreenDark | Lime | Yellow | Amber |
| AmberDark | Orange | OrangeDark | Brown | Blu… |
| Grey50 | Grey100 | Grey200 | Grey300 | Grey… |
| Grey | Grey600 | Grey700 | Grey800 | Grey900 |

FromHex("#3F51B5")

Back to TOC

## 25. Optional-content layers (OCG)

Document.WithLayer(name) registers an OCG that PDF viewers expose in the layers panel. Wrap any slot's chain in **.Layer(handle)** to scope its draws into that layer — the viewer can toggle it without regenerating the document.

| | | |
|---|---|---|
| Always visible (no .Layer wrap) | Design layer (visible by default) | Notes layer (hidden by default) |

*Open the layer panel in your PDF reader to toggle visibility.*

Back to TOC

## 26. Multi-page FluentDocument.Create

A single **FluentDocument.Create** callback can call **.Page(...)** multiple times — each is an independent paginating block with its own size, margins, and slots. Page numbers count across blocks. This document uses one main block + an appendix block at the end.

Back to TOC

## 27. Document config: Outline, Conformance, PageLabels

Three document-level builders feed catalog entries the layout engine doesn't see directly:

**.WithOutline(o => ...)** — bookmark tree referencing Section anchors. This document's outline lists each major section; open the outline panel in your reader to see it.

**.WithConformance(Conformance.PdfA1b)** — declare PDF/A-1b or PDF/UA-1 conformance; XMP namespaces and catalog entries are emitted automatically. Pair with *.WithLanguage("en-US")* for PDF/UA. (This document is not declared conforming because it intentionally exercises the DRAFT watermark transparency, but the API is wired in.)

**.WithPageLabels(pl => pl.Range(0, ...).Range(2, ...))** — viewer-displayed page numbers like *i, ii, 1, 2* or *A-1, A-2*. This document labels its appendix with prefix "A-".

Back to TOC

Back to TOC

## Appendix — second Page block, landscape orientation

# Why two Page blocks?

A Document can mix orientations and page sizes by using multiple .Page(...) blocks. Page numbers are continuous: this is the first appendix page, and TotalPages includes both blocks.

Showcased so you can verify the renderer's two-pass count handles blocks of different dimensions correctly.