



The soli-pdf template

A complete, worked reference — rendered by the engine it documents.

- every element & option
- each shown with its JSON
- one template, one call

This document is itself a soli-pdf template. Everything you see — the cover band, the tables, the charts, the QR code — was produced by a single JSON template passed to one `pdf_render()` call, with no browser and no HTML. Each feature below is shown twice: the JSON that declares it, and the result it renders to.

CONTENTS

1	Text & flow — paragraphs, spans, lists	p. 2
2	Graphics & codes — rects, strokes, images, QR, barcodes	p. 4
3	Tables — data binding, merges, rich cells	p. 6
4	Charts — bar, line, pie, donut	p. 8
5	Layout & render options — columns, branches, the render call	p. 11

solilang.com/docs/builtins/pdf · [edit this template live in the playground](#)



1

Text & flow

Paragraphs, inline rich text, and lists — the words on the page.

paragraph

The workhorse: a wrapped block of text. Options set size, weight, colour and alignment.

```
{ "type": "paragraph",  
  "value": "Invoice ${invoice.number}",  
  "options": { "fontSize": 16, "fontWeight": "bold", "color": "0F766E" } }
```

renders

Invoice INV-2026-0042

alignment & justify

alignment is left · center · right · justify. justify spreads each full line to both margins by widening word gaps; the last line stays left.

```
{ "type": "paragraph", "options": { "alignment": "justify" },  
  "value": "soli-pdf typesets text itself..." }
```

renders

soli-pdf typesets text itself — measuring, wrapping and justifying every line in-process, with no headless browser in sight. This paragraph is set justify, so both edges line up while the final line stays left.

lineHeight & spacing

lineHeight multiplies the leading (engine default 1.2); spacing adds points below the block, replacing a trailing move.

```
{ "type": "paragraph",  
  "options": { "lineHeight": 1.7 },  
  "value": "Loose leading lets dense reference text breathe." }
```

renders

Loose leading lets dense reference text breathe — this block is set at lineHeight 1.7, so the wrapped lines sit further apart than the 1.2 default.



spans – inline rich text

spans mix weight, size, colour, links, underline and strike within one wrapped flow; each span inherits the paragraph options it omits.

```
{ "type": "paragraph", "spans": [
  { "text": "Mix " }, { "text": "bold", "fontWeight": "bold" },
  { "text": ", a ", }, { "text": "link", "link": "https://solilang.com",
    "color": "2563eb", "underline": true },
  { "text": ", and a redline was " },
  { "text": "99.00", "strike": true, "color": "b91c1c" },
  { "text": " now 79.00", "fontWeight": "bold", "color": "0F766E" } ] }
```

renders

Mix **bold**, *italic*, mono, a [link](https://solilang.com), and a redline was ~~99.00~~ now **79.00 EUR**

list – ordered • nested • custom marker

ordered numbers from start; items may nest a list or carry spans; marker and indent tune the look.

```
{ "type": "list", "ordered": true, "items": [
  "Grind the beans",
  { "text": "Brew", "list": { "items": ["Bloom 30 s", "Pour to 250 g"] } },
  { "spans": [ { "text": "Enjoy " },
  { "text": "responsibly", "italic": true } ] } ] }
```

renders

1. Grind the beans
2. Brew
 - Bloom 30 s
 - Pour to 250 g
3. Enjoy *responsibly*



2

Graphics & codes

Boxes, strokes, images, and machine-readable codes.

move – the cursor primitive

The layout primitive. +y is down, -y up, +x right. rect, line, ellipse, image, qr and barcode draw at the cursor without advancing it — move positions them (that is how these result rows are placed).

```
{ "type": "move", "x": 20, "y": -8 }
```

rect – panels & boxes

A filled and/or stroked box. radius rounds the corners; dash strokes the border. Drawn at the cursor, never advancing it.

```
{ "type": "rect", "width": 495, "height": 44,  
  "fill": "f0fdfa", "border": "99f6e4", "radius": 10 }
```

renders

rounded • filled • bordered

"dash": [5, 3]

ellipse • line • hr – strokes & dots

ellipse is a circle when rx == ry; line runs from the cursor to +(dx, dy); hr rules across the width (or an explicit width). line & hr take a dash too.

```
{ "type": "ellipse", "rx": 6, "ry": 6, "fill": "0F766E" }  
{ "type": "line", "dx": 170, "dy": 0, "dash": [5, 3] }  
{ "type": "hr", "color": "f59e0b", "thickness": 1.4, "width": 220 }
```

renders





image – raster & SVG

PNG · JPEG · WebP · GIF and SVG (auto-rasterised, so vector logos stay crisp). width alone derives the height from the aspect ratio.

```
{ "type": "image", "width": 150,  
  "value": "data:image/svg+xml,<svg ...>soli-pdf</svg>" }
```

renders

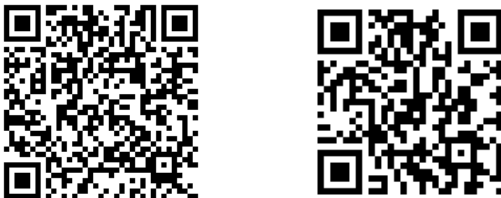


qr – scan-to-pay & text

kind "epc" builds a SEPA GiroCode from the payment data — scan it in a banking app to pre-fill the transfer; kind "text" encodes any string. Both rasterised locally, no network.

```
{ "type": "qr", "kind": "epc", "width": 88,  
  "name": "${payment.name}", "iban": "${payment.iban}",  
  "amount": "${payment.amount}", "currency": "EUR",  
  "remittance": "${doc.ref}" }
```

renders



left: EPC scan-to-pay (79.00 EUR to Soli Demo SARL) · right: a URL

barcode – code128 · ean13

1-D barcodes. code128 takes any printable ASCII; ean13 takes 12 digits and computes the check digit. humanReadable prints the caption under the bars.

```
{ "type": "barcode", "symbology": "code128", "value": "${doc.ref}",  
  "width": 210, "height": 46, "humanReadable": true }
```

renders



TOUR-2026-001



400638133393



3 Tables

Grids of cells — data-bound rows, merges, rich content, and per-table stamps.

table — data binding · header · stripe

data binds the template row to an array, repeating it per item. The header_columns band repeats on every page; options.stripe zebra-fills every second body row.

```
{ "type": "table", "data": "items",  
  "header_columns": [ { "text": "SERVICE", "width": 250 }, ... ],  
  "rows": [ [ { "text": "${name}", "width": 250 }, ... ] ],  
  "options": { "header": { "fillColor": "0F766E", "textColor": "FFFFFF" },  
    "stripe": "f1f5f9" } }
```

renders

SERVICE	QTY	AMOUNT
Template engine	1	890.00 EUR
Chart pack (bar · line · pie · donut)	4	450.00 EUR
QR & barcode primitives	2	300.00 EUR
Factur-X / PDF-A-3b embedding	1	300.00 EUR

colspan & per-cell fill — a totals row

colspan merges a cell across column slots; a cell's own fill paints its background over the stripe — the recipe for summary and total rows.

```
{ "rows": [ [  
  { "text": "TOTAL DUE", "colspan": 2, "alignment": "right",  
    "fontWeight": "bold" },  
  { "text": "${doc.total}", "alignment": "right", "fill": "fef3c7" } ] ] }
```

renders

TOTAL DUE	1,940.00 EUR
-----------	--------------

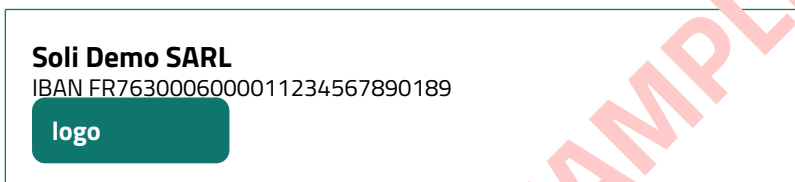


rich cells · borderSides · per-table watermark

A rich cell stacks text lines and an image in one cell; borderSides sets each edge with a borderColor; the table carries its own centred watermark.

```
{ "type": "table",  
  "watermark": { "text": "SAMPLE", "fontSize": 44, "color": "fecaca" },  
  "rows": [ [ { "content": [ { "type": "text", ... },  
    { "type": "image", ... } ] }, ... ] ] }
```

renders



valign "bottom"

repeat – block-level data binding

The block-level analogue of a data-bound row: lay out any elements once per array item, with `${field}` scoped to each item.

```
{ "type": "repeat", "data": "milestones", "content": [  
  { "type": "paragraph", "spans": [  
    { "text": "${date} ", "mono": true, "color": "64748b" },  
    { "text": "${title}", "fontWeight": "bold" },  
    { "text": " – ${note}" } ] } ] ] }
```

renders

- 2026-03 **Charts & lists** — bar, line, pie, nested lists
- 2026-06 **SVG & barcodes** — crisp vector logos, Code 128 / EAN
- 2026-07 **Flow & polish** — columns, tokens, the `${` escape



4 Charts

Bar, line, pie and donut — drawn from the data, no plotting library.

chart — grouped bars • multi-series • gridlines

values is an array of { field, name, color }: each series reads its own field, bars group per category, and the legend lists the names. gridlines adds the value axis.

```
{ "type": "chart", "kind": "bar", "data": "quarters", "label": "q",  
  "gridlines": true, "width": 495, "height": 150,  
  "values": [ { "field": "fy25", "name": "FY 2025", "color": "94a3b8" },  
    { "field": "fy26", "name": "FY 2026", "color": "0F766E" } ] }
```

renders

Grouped — one bar per series

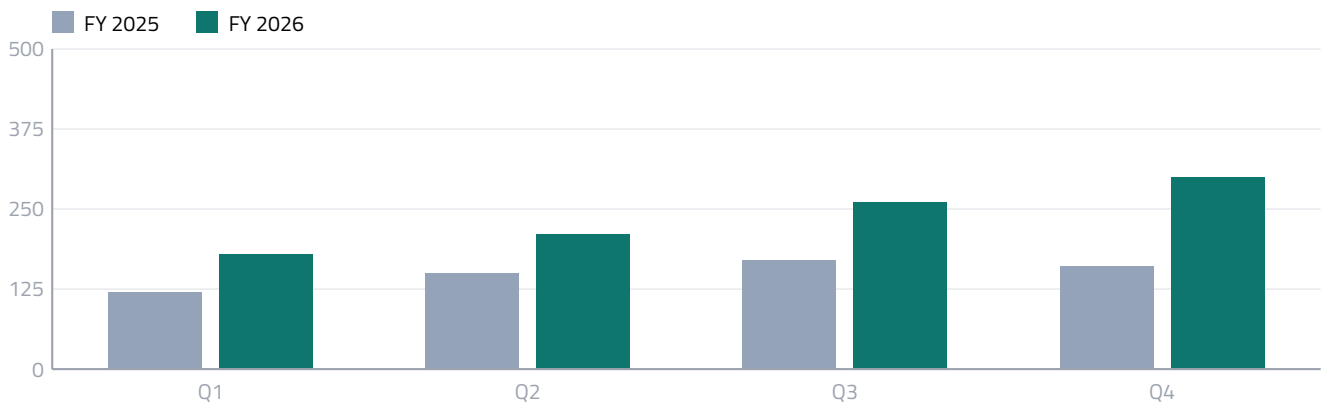




chart – stacked bars

mode "stacked" piles the series per category instead of grouping them; a colors palette is cycled when series set no colour of their own.

```
{ "type": "chart", "kind": "bar", "mode": "stacked",  
  "data": "quarters", "label": "q", "gridlines": true,  
  "colors": [ "0F766E", "f59e0b" ],  
  "values": [ { "field": "fy25" }, { "field": "fy26" } ] }
```

renders

Stacked — series pile per category

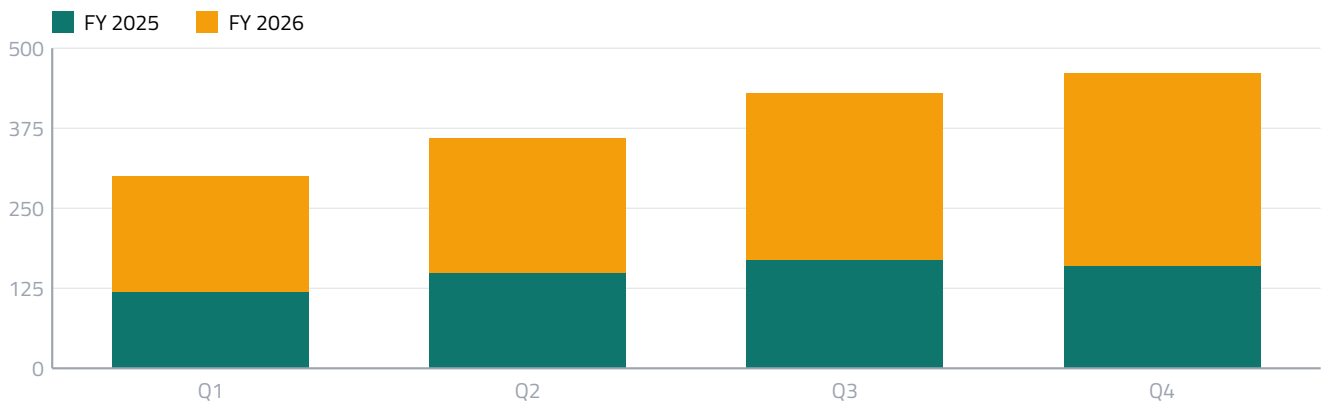


chart – line

One line per series over the category axis; share the same data binding as the bar charts.

```
{ "type": "chart", "kind": "line", "data": "quarters", "label": "q",  
  "width": 495, "height": 120,  
  "values": [ { "field": "fy26", "name": "FY 2026", "color": "0F766E" } ] }
```

renders

FY 2026 by quarter

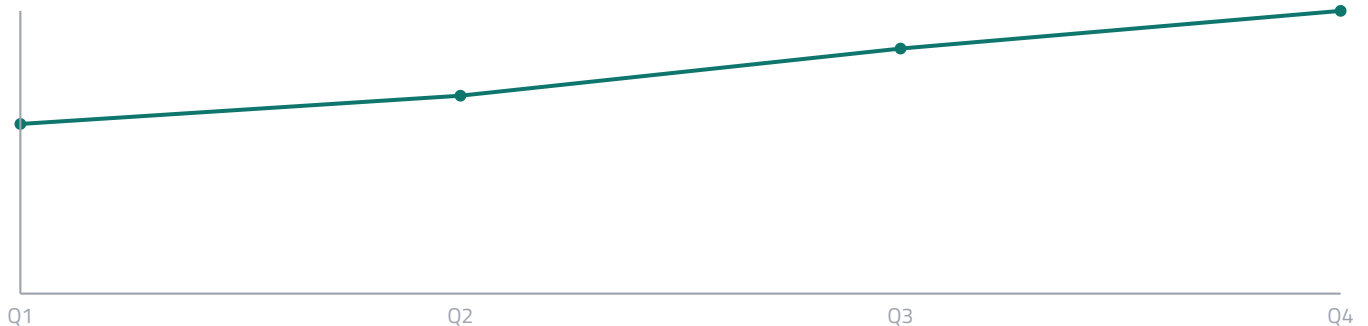




chart – pie & donut

points may be inline instead of data-bound. pie draws a swatch + percentage legend; donut is a pie with a ring cutout ("kind": "donut").

```
{ "type": "chart", "kind": "pie", "width": 240, "height": 130,  
  "points": [ { "label": "Templates", "value": 45 },  
    { "label": "Charts", "value": 25 }, ... ] }
```

renders

Pie



■ Templates 45%
■ Charts 25%
■ Codes 18%
■ Everything else 12%

Donut



■ Templates 45%
■ Charts 25%
■ Codes 18%
■ Everything else 12%



5

Layout & render options

Multi-column flow, data-driven branches, and the options that live in the render call rather than the template.

columns – multi-column flow

Children fill column 1 to the bottom, then column 2 (sequential fill); a `page_break` inside is a column break. Everything flows — paragraphs, lists, images, tables and charts. A table that overflows a column continues in the next one with its header repeated.

```
{ "type": "columns", "count": 2, "gap": 22, "content": [
  { "type": "paragraph", "value": "flows down column 1..." },
  { "type": "table", "header_columns": [...], "rows": [...] } ] }
```

renders

Column one fills top to bottom, then the flow hops into column two automatically — sequential fill. A table renders inline too:

Item	Qty
Widgets	12
Gadgets	5
Sprockets	9

...and the paragraph after the table keeps flowing, spilling into the second column to show the block is genuinely two columns.

if / unless – data-driven branches

Render content only when a condition holds (if) or fails (unless); an optional else is the other branch. With equals it is a string test, otherwise a truthiness test.

```
{ "type": "if", "when": "doc.paid", "equals": "true",
  "content": [ { "type": "paragraph", "value": "PAID IN FULL" } ],
  "else": [ { "type": "paragraph", "value": "Balance due" } ] }
```

renders

PAID IN FULL — `doc.paid` is "true", so the content branch renders (the else branch is skipped).



interpolation & page tokens

A dotted path pulls from the data; page tokens resolve after pagination and turn a contents list into real page numbers. Print a literal brace token by doubling the dollar.

```
"Invoice ${invoice.number} — page #PAGE# of #TOTAL_PAGE#"
"See charts on p. #PAGE_OF:sec-charts#" with "linkTo": "sec-charts"
"${ prints a literal $" — this document's code boxes rely on it
```

renders

The running header and footer of this page both resolve `#PAGE#` / `#TOTAL_PAGE#`, and the cover's contents used `#PAGE_OF:...#` for its page numbers.

document options

The options block sets page size, margins, the reserved header_height, a page background or backgroundImage (this cover's wash — opacity fades it to a soft tint), a document watermark, and tagged for accessible output.

```
"options": {
  "page": "a4", "margins": { "top": 42, "left": 50 },
  "header_height": 34,
  "watermark": { "text": "DRAFT", "front": true, "pages": "first" },
  "backgroundImage": { "src": "...", "pages": "first",
  "opacity": 0.15 } }
```

the render call — pdf_render(tpl, data, options)

Some features live in the call, not the template JSON: a letterhead to draw on, files to embed, Info-dictionary metadata, a password, or PDF/A archival output.

```
pdf_render(tpl, data, {
  "stationery": "pdf/letterhead.pdf", # draw on a letterhead PDF
  "attachments": [{ "path": "data.csv" }],
  "title": "Invoice 42", "author": "Soli",
  "font_dirs": ["font"], "fetch_images": false })
```

Built with Soli — one JSON template, one pdf_render() call.

solilang.com/docs/builtins/pdf · try this very template live in the playground