Renderer Sampler

A long-form stress test for tables, lists, and the things in between

This document exercises every element the faux-press identity-preserving renderer is supposed to set correctly. Open it in the editorial-essay preset to see the production target; open it again in book-chapter or technical-manual to see how the same source survives a change of voice. Nothing here is randomly generated. Every paragraph, list, and table was written to flush out a specific class of regression — wrap thresholds, column width drift, list-marker indent, footnote routing, header repetition, baseline grid alignment.

A good renderer should make this document feel inevitable. A merely correct renderer makes it readable. The interesting tests are the ones that ride the boundary between the two.

1. Plain prose

A paragraph of plain prose, set in body type, with no inline decorations. The line-breaker has to find graceful breaks across the entire measure, hyphenating where appropriate and leaving the right amount of slack on the last line so the paragraph does not finish on a runt. If your output here looks tight on some lines and loose on others, the Knuth–Plass solver is not converging — check the badness threshold and the tolerance ladder.

The second paragraph appears immediately below, with a first-line indent if the preset is BookChapter and a vertical gap if the preset is EditorialEssay. The shape of the page boundary between these two paragraphs is the most-tested decision in book typography. There is more disagreement about how to separate paragraphs than about how to set words inside them.

A third paragraph, this one with *emphasis* and **strong emphasis** and a `monospace fragment` and a deliberate ellipsis at the end so the trailing dot spacing can be inspected…

2. Headings

You have just rendered an `h2`. Above this paragraph there is an `h3`:

A heading at level three

Followed by a paragraph in body type. The vertical rhythm between heading and paragraph should be consistent with the rest of the document; if it looks tighter or looser than the same gap between two body paragraphs, the heading-to-paragraph cascade has a stale entry.

Level four

A paragraph after the level-four heading. Headings at level four and below often share styles with level three in book contexts; that is a deliberate cascade flattening, not a bug.

Level five

A paragraph after level five.

Level six

A paragraph after level six.

3. Unordered lists

A short unordered list:

- Apples
- Oranges
- A longer item that runs to two visual lines so the list-marker indent can be verified — the marker should stay vertically aligned with the first character of the first line, and the second line should sit flush with the body indent.
- Pears

A nested unordered list:

- Top-level item one
  - First nested item
  - Second nested item
    - Deeper still

        - And again, with more text so we can see the indent grow predictably
    - Back up one
- Top-level item two
- Top-level item three, with a particularly long body so we can confirm wrap behaviour at the second-level indent. The visual measure for this paragraph is the full body width minus the cumulative left indent of every list level above this one.

4. Ordered lists

A short ordered list:

1. First
2. Second
3. Third

A nested ordered list inside an unordered list:

- Steps to compose a paragraph:
    1. Tokenize source text into runs.
    2. Shape each run at the source font.
    3. Compute candidate line breaks via Knuth–Plass.
    4. Score breaks by demerits.
    5. Select the lowest-demerit candidate set.
- Quality checks:
    1. No runt last lines.
    2. No widows or orphans.
    3. No hyphen-stack of length greater than two.

An ordered list with multi-paragraph items:

1. The first item has a single short paragraph.
2. The second item has two paragraphs.

    This is the second paragraph of item two. It should be indented to align with the first paragraph, sitting under the same body measure with the marker absent.
3. The third item has a paragraph and then a nested unordered list.

- Note A
- Note B
- Note C

4. The fourth item contains a paragraph followed by a code fragment:

```rust
fn knuth_plass(items: &[Item], measure: f64) -> Vec<Line> {
    // Pure shortest-path-on-DAG demerit search.
    solve_items(items, measure)
}
```

   And a closing sentence in the same item.

5. Task lists

The renderer should treat task markers as identity-preserving decorations, not as text inside the body.

- ☑ Define the per-cell measurement contract.
- ☑ Wire the water-filling column solver.
- ☑ Resolve table axis defaults from the active `ThemePreset`.
- ☐ Thread `--preset` from CLI through `LayoutMeasurementProfile`.
- ☐ Hand off responsive verdict to the cell vector transformer.

6. Block quotes

A short block quote:

> Typography is the visual translation of language. Reading is a long, repeated, intimate engagement; setting a paragraph is a one-time decision that shapes that engagement forever.

A multi-paragraph block quote:

> First paragraph of the block quote. The left rule (if the preset draws one) should run unbroken across the paragraph boundary.
>
> Second paragraph of the block quote. The vertical rhythm inside the quote should be the same as outside; quotes are *containers*, not *new typographic worlds*.

A block quote with a nested list:

> A short list inside a quote:
>
> - One
> - Two
> - Three

7. Inline code and code blocks

Inline code samples: the function `compose_paragraph` lives in `crates/-paragraph-composer`. Its primary type, `ParagraphSetterProfile`, carries fields like `measure_em` (a floating-point value) and `hyphenation` (an enum). Mixing inline code with prose like this should not affect the surrounding body's baseline grid.

A fenced code block with a language tag:

```Rust
pub fn solve_table_layout_auto_with_policy(
    cells: &[CellMeasurement],
    column_count: usize,
    available_inline_pt_milli: i32,
    cell_font_size_pt_milli: i32,
    cell_inline_padding_pt_milli: i32,
    sizing_mode: TableSizingMode,
    excess_allocation: TableExcessAllocation,
) → (Vec<i32>, TableColumnWidthDerivation) {
    // ...
}
```

A second fenced code block in a different language:

```Python
def water_fill(columns, available):
    """Each column gets fair_share = remaining / unsettled;
 happy cols cap at max."""
    columns.sort(key=lambda c: c.max)
    remaining = available
    for col in columns:
        fair_share = remaining / (len(columns) -
columns.index(col))
        if col.max ⩽ fair_share:
            col.width = col.max
            remaining -= col.max
        else:
```

```
            col.width = fair_share
            remaining -= fair_share
```

A code block with longer lines, which should wrap or clip according to the preset's `CodeListing` policy:

```
This is a deliberately long line in a code block to verify that line wrapping is disabled in code listings and that the overflow policy emits a horizontal-overflow diagnostic when the line exceeds the measure.
```

8. Tables — the heart of the test

A small one-row, three-column table:

ID	Name	Status
001	Widget	Active

A larger table with multiple body rows and a numeric column with right alignment + tabular figures:

ID	Item	Quantity	Unit Price	Total
001	Widget	12	9.99	119.88
042	Gadget	3	12.50	37.50
113	Doohickey	40	3.75	150.00
256	Whatchamacallit	1	18.00	18.00
314	Thingamajig	5	7.25	36.25

A table with one wrapping prose column:

ID	Description
1	A short note.
2	A deliberately long paragraph constructed to wrap past the default widow/orphan threshold of four lines at any plausible column width. We pad with additional clause-rich prose so the wrap count remains robust across measure choices, and the row-height aggregator picks up the multi-line shaped height.

ID	Description
3	Another short.
4	A second long paragraph, this time about a different topic, to give the row-fragment fitter a second case to chew on: page splits should respect line boundaries inside the cell, and never pixel-slice a glyph cluster.

A six-column table that will exercise the responsive policy ladder at narrow targets — and the water-filling solver at the default body:

Code	Name	Region	Status	Owner	Updated
A001	North alpha	Northeast US	Active	Mira Sen	2025-01-15
A002	North beta	Northwest US	Active	Joel Park	2025-01-12
B003	South gamma	Southeast US	Pending	Anika Doe	2025-01-10
B004	South delta	Southwest US	Inactive	Joel Park	2024-12-30
C005	Mountain epsilon	Mountain West	Active	Mira Sen	2025-01-08
D006	Coastal zeta	Pacific NW	Active	Anika Doe	2025-01-04

A table with an empty cell, to verify the row-min-height policy:

Col A	Col B	Col C
one		three
four	five	six

A two-row table with a long cell that should still keep the header tightly stacked on the body row above it (no header-orphan):

Key	Value
description	A long-form value cell that exercises wrapping inside a two-column key/value table. The key column is narrow; the value column absorbs all the slack. With the right preset, the value column should wrap to multiple visual lines while the key stays on a single line, top-aligned with the first line of the wrapped value.
status	OK

9. Inline elements inside tables

A table whose cells contain inline emphasis, code, and links:

Feature	Renderer State	Notes
`tnum` on digits	**Wired** — Phase 8.4.4	Numeric / Date columns only
Drop cap	*Editorial only*	Falls back to plain at 9pt body
Hyphenation	`EnglishUsHyphenator`	Liang–Knuth tables
Span repair	**Wired** — Phase 8.10.1	Proportional row-height fan-out

10. Footnotes

A paragraph with a single footnote reference, just for the line-shaping test.[1] The reference number should sit superscript and not affect the visual baseline of the surrounding text.

A second paragraph with a different footnote.[2] The chapter-composition solver routes these to the page where their reference appears; multi-fragment tables route to the fragment's page.

A paragraph with two footnote references in close proximity:[3] [4] this is the kind of construction that historically broke layout because the second reference's reservation block conflicted with the first.

A footnote reference inside a table cell:

Subject	Notes
Topic A	A short observation.[5]
Topic B	Another short observation.[6]

1. The first footnote body. Short, declarative, and traceable back to its anchor.
2. The second footnote body. The chapter solver routes this to the page where the anchor lands; the reservation budget is computed before pagination, so the page break is honest about footnote weight.
3. Footnote three, paired with footnote four in the same paragraph. The pagination layer must not produce a duplicate area reservation.
4. Footnote four. When this and three are on the same page, the area packs both bodies without re-paying the area-height tax twice.
5. Per-cell footnote. The pagination layer routes this body to whichever page the cell's row lands on (Phase 8.12).
6. Second per-cell footnote. Two anchors in the same cell would still route to the same page; two anchors in different cells of the same row also route to that row's page.

11. Images

The renderer treats images as atomic inline-replaced content. When the image has an intrinsic resolution and the available inline is narrower than the natural width, the image scales down with aspect ratio preserved.

A placeholder figure with descriptive alt text.

The above reference points at a missing file on purpose — the renderer's image-resource resolver should emit a typed loss record (`ResourceKind::RasterImage` unavailable) rather than panicking. The surrounding paragraph wraps as if the image's content area were the configured fallback rectangle.

12. Mixed nesting

A bulleted list with a table inside one of the items:

- Quarterly sales

Quarter	Units	Revenue
Q1	1200	$14,400
Q2	1450	$17,400
Q3	1100	$13,200
Q4	1675	$20,100

- Annual headline: $65,100 across 5,425 units, with the Q4 surge accounting for thirty-one percent of revenue.
- Closing remarks paragraph that wraps across multiple visual lines to confirm the indent stays consistent with the bulleted item indent above the table, and the table sits flush with the body of the list item.

An ordered list with a quote inside an item:

1. Premise:

> Identity must be preserved through every transformation, even when freedom is lost.

2. Corollary: any responsive policy that discards a column must record the loss; never silent.
3. Consequence: the IR carries `LossTranslationNodeInstance` alongside every responsive fire.

13. Long-form prose to drive page breaks

A short story-shaped section to push the renderer past a page break and watch the running matter behave.

The conference had been advertised as a debate about the future of typesetting, and Linnea had spent two weeks preparing a talk that her co-panelists would later describe as the most controversial of the year. She did not think it was controversial. She thought it was obvious. Identity preservation was not a fashion; it was the precondition for any honest rendering pipeline. If you could not trace a glyph on a page back to the byte in the source that produced it, you had not rendered the document, you had merely produced something that looked like it.

The first audience question, predictably, was about performance. *Why*, asked a serious young man in the front row, *should anyone pay the cost of carrying provenance through every IR layer when the user only sees the final pixels?* Linnea had been asked this question so many times that her answer had developed a polish she now regretted. *Because*, she said, *the pixels are not the contract. The contract is "this is what the author wrote, set in this typeface, at this measure, with these breaks." The pixels are evidence of the contract being honored. Without provenance, you can produce a pretty page that is also a lie.*

Someone in the third row applauded, then stopped when she realized no one was applauding with her. The panelist to Linnea's right — an engineer from a much-larger company whose products produced pixels at scale without any auditing of what they had become — leaned into his microphone and said, with the polite contempt that engineers learn at conferences, *we have not yet had a customer ask for what you are describing.*

Linnea took a slow breath. She was tired. Two weeks of preparation, three hours of sleep, and the certainty that this was going to be a long evening. *Of course*, she said. *Customers do not ask for what they cannot imagine. The job of the renderer is not to satisfy the imagination of the customer. The job of the renderer is to be honest.*

The conversation continued, predictably, into edge cases and trade-offs. Someone asked about table layout. Someone else asked about footnotes. A third person asked whether the proposed identity invariant was compatible with multi-column setting. Linnea answered each question without losing her temper, because losing one's temper at conferences was, she had learned, a more reliable way to lose an argument than being wrong.

By the time the moderator cut them off for the next session, Linnea was already drafting the next chapter of her book in her head. *Identity*, she would write, *is not a feature. It is the substrate on which every other feature stands.*

She walked back to her hotel through a city that had been redesigned by people who had not consulted her, and she allowed herself, for the length of one slow block, to imagine that this might one day be different.

14. Closing

The end of the sampler. If you have read this far, the renderer has shaped, paginated, and emitted at least fourteen sections covering plain prose, six levels of heading, ordered and unordered lists with nesting, task lists, block quotes, inline and fenced code, tables of varying sizes and column compositions, footnotes anchored in prose and in cells, images with deliberately-missing resources, lists containing tables, ordered lists containing quotes, and a long block of narrative prose for pagination stress.

Any visible regression is a regression. Open the IR derivation trace via `arch report --json` and walk back from the offending placement to find which layer of the pipeline lost its honesty.