The mental model

Your intuition is mostly right, with one nuance worth pinning down.

The renderer separates **what your document is** from **how it looks**:

- **Semantic structure** comes from your Markdown plus the book project layout (`book.toml`, `SUMMARY.md`, chapter files). Headings become chapters and sections, blockquotes become quotations, fenced code becomes listings, footnote markers become footnote references. These are *publication roles* — the renderer doesn't let you "configure them away," they're what the document **is**.
- **Appearance** is everything else: page geometry, fonts, sizes, colors, spacing, paragraph behavior, justification, hyphenation, where chapters start, how blockquotes look, how tables break, whether code is highlighted.

What's slightly more nuanced than "themes for appearance": there are **four orthogonal axes** that all affect appearance and behavior, plus a **design bundle** that ties them together.

```
                ┌─ target ──────  page geometry (A4, trade paperback, …)
                ├─ theme ───────  visual taste (palette + font families)
design bundle ──┤
                ├─ policy pack ─  behavior (justified, footnote placement, …)
                └─ component kit  component variants (blockquote, code, table)
```

A **design** is a named bundle that picks values along each axis and then adds finer-grained **tokens** (specific font size, exact color, exact margin) and **policies** (specific paragraph style, exact heading break behavior). Builtin designs ship with the renderer. Your own designs are written as YAML files that *extend* a builtin and override what you care about.

So the cleanest restatement of your model:

- **Markdown** + **book project** decides *semantic structure*.
- **Builtin designs** ship sensible appearance defaults for common shapes (novel, technical manual, academic paper, etc.).
- **Your design YAML** layers your own appearance overrides on top.

How to render a book

Two entry points:

```
# Single Markdown file.
faux-press render <file.md> \
  --design file:my-theme.yaml \
  --target print-a4 \
  -o out.pdf

# Multi-file book project with book.toml + SUMMARY.md.
faux-press book render <project-root> \
  --design file:themes/warm-rail.yaml \
  --target print-a4 \
  -o out.pdf
```

`<project-root>` is the directory that contains `book.toml`. The `src/` (or whatever `book.toml` points to) contains `SUMMARY.md` and the chapter `.md` files. The renderer reads `[output.faux]` in `book.toml` for defaults; the CLI flags override them.

Useful render flags

Flag	What
`--design <ref>`	`builtin:<name>`, `file:<path.yaml>`, or `user:<id>`
`--target <name>`	Page geometry (see *Targets*)
`--mode <mode>`	`identity` (default, full quality), `preview-fast`, `balanced`, `print-final`, `exhaustive-debug`
`--fast`	Skip metadata records (PDF is byte-identical, ~200 ms faster)
`--fast-pdf`	Skip DEFLATE on content streams (larger file, faster emit)
`--trace`	Emit cascade trace alongside output
`--chapter-level h2`	Override what heading level starts a chapter

Other commands worth knowing

Command	What
`faux-press explain <key> <file>`	Show how a settings value was resolved
`faux-press lint <file>`	Static typography lints
`faux-press inspect <file>`	Dump structure (publication tree, roles)
`faux-press new-preset <template>`	Scaffold a design YAML
`faux-press doctor`	Check fonts, tools, environment
`faux-press proof`	Emit per-decision debug artifacts
`faux-press design list` / `design show`	Browse builtin designs
`faux-press tool`	JSON envelope mode for MCP/agent wrappers

book.toml

A minimal multi-file book project:

```toml
[book]
title = "Keeping Your Word"
authors = ["Saurabh Suman"]
description = "Why reliability is the rarest skill, and how to build it."
language = "en"
src = "book"

[output.faux]
design = "file:themes/a4-warm-rail.yaml"   # or "builtin:book-novel"
target = "print-a4"
```

`src/SUMMARY.md` is the same mdBook format the rest of the ecosystem uses (nested bullet lists pointing at `.md` files). Faux-press reads it; chapter order, part grouping, and appendix grouping come from it.

Designs

A design is a YAML file with the schema `faux.design/v1`. It extends one builtin (or another file), then layers in any of: `intent`, `tokens`, `policies`, `components`, plus optional per-target overrides.

Minimal valid design:

```yaml
schema: faux.design/v1
id: kasa.warm-novel
version: 0.1.0
name: Kasa Warm Novel
extends:
  - builtin:book-novel

tokens:
  typography.body.family: Fraunces
  typography.body.size: 11.6pt
```

A larger one, with overrides on the appearance and prose behavior:

```yaml
schema: faux.design/v1
id: kasa.warm-rail
version: 0.1.0
name: Warm Rail
extends:
  - builtin:book-novel

intent:
  kind: manual
  target: print-a4
  theme: warm-literary
  policyPack: book-print
  componentKit: literary-minimal
  quality: publish

tokens:
  page.margin.top: 82pt
  page.margin.right: 76pt
  page.margin.bottom: 102pt
  page.margin.left: 138pt
  typography.body.family: Fraunces
  typography.body.size: 11.6pt
  typography.body.leading: 1.50
  typography.body.first_line_indent: 1.15em
  typography.heading.family: Fraunces
  typography.heading.weight: 675
  typography.code.family: Fira Code
  typography.code.size: 9.45pt
```

```
  color.text: "#151716"
  color.heading.text: "#080d0f"
  color.quote.rule: "#b64631"
  color.code.background: "#eeede9"
  color.code.text: "#1b1d1e"

policies:
  paragraphs.style: indented-prose
  paragraphs.paragraph_gap: 0pt
  publication.roles.chapter.page_start: new-page
  publication.roles.chapter.opening_treatment: chapter-opener
  publication.roles.section.page_start: flow
  typography.justification: justified
  typography.hyphenation: liang-english-us
  typography.inline_harmony.enabled: true
```

Design references

Form	Where it resolves
`builtin:<name>`	Built-in design (see *Builtin presets* below)
`file:<path.yaml>`	YAML file on disk
`user:<id>`	Design installed in your user design registry

Extension and override semantics

Designs cascade. When `extends: [builtin:book-novel]` is set, the parent's tokens and policies are applied first; your file's values overwrite. The final resolution order is roughly:

```
defaults
  → target preset
    → theme preset
      → policy-pack preset
        → component-kit preset
          → parent design (recursively)
            → your design
              → book.toml [output.faux]
                → CLI flags
                  → document frontmatter
```

You can override the same key at any layer; the latest one wins. Use `faux-press explain <key> <input>` to see the path.

Token reference

Tokens are typed visual variables. Every token has a stable kebab-case key, a value type, a default value, and a list of pipeline stages that consume it. The current registry is checked into `crates/render-settings/src/lib.rs` under `token_registry()`.

Typography

Token	Type	Notes
`typography.body.family`	font family	any Google Fonts name
`typography.body.size`	length	e.g. `11.4pt`
`typography.body.leading`	unitless	`1.45`, `1.50`
`typography.body.first_line_indent`	length	`1.15em`
`typography.heading.family`	font family	
`typography.heading.weight`	int 100–900	`600`, `675`, `700`
`typography.code.family`	font family	`Fira Code`, `IBM Plex Mono`
`typography.code.size`	length	`9.4pt`
`typography.caption.size`	length	
`typography.footnote.size`	length	
`typography.footnote.leading`	unitless	
`typography.inline_harmony.max_scale_up`	unitless	inline-code vs body x-height correction max
`typography.inline_harmony.max_scale_down`	unitless	(lower bound)
`typography.initial_letter.book_opening.size_lines`	int	drop cap height in lines
`typography.initial_letter.book_opening.sink_lines`	int	drop cap sink
`typography.initial_letter.book_opening.gap_inline_end`	length	

Color

Token	What
`color.page.background`	full-bleed page background
`color.text`	body text
`color.heading.text`	heading text
`color.initial_letter.text`	drop cap color
`color.quote.rule`	blockquote rule
`color.code.background`	code block fill
`color.code.text`	code text
`color.code.gutter`	gutter (line-number column) fill
`color.code.line_number`	gutter glyphs
`color.code.rule`	gutter rule
`color.code.keyword`	syntax highlight
`color.code.string`	
`color.code.comment`	
`color.code.literal`	
`color.table.rule`	
`color.table.cell_background`	
`color.table.header_background`	
`color.table.alt_row_background`	
`color.table.hairline`	
`color.footnote.text`	
`color.footnote.rule`	
`color.link.text`	
`color.generated.text`	generated text (folios, “Chapter N”, etc.)

All colors accept `#rrggbb`. Dark mode is just "set `color.page.background` to something dark and adjust the rest."

Spacing

Token	What
`spacing.paragraph.gap`	between paragraphs (only used when `paragraphs.style: spaced-prose`)
`spacing.section.clearance`	clearance around section heads
`spacing.heading.before.h1` / `.after.h1`	vertical space around h1
`spacing.heading.before.h2` / `.after.h2`	…h2
`spacing.heading.before.h3` / `.after.h3`	…h3
`spacing.quote.inset`	inset from outer column
`spacing.quote.before` / `.after` / `.between_series`	
`spacing.code.before` / `.after`	
`spacing.code.padding.block` / `.padding.inline`	
`spacing.table.before` / `.after` / `.row.block` / `.row_gap` / `.cell.block` / `.cell.inline`	
`spacing.list.item_gap` / `.before` / `.after` / `.marker_gap`	
`spacing.footnote.marker_column` / `.marker_gap`	
`spacing.section.break_before`	extra space before a new section

Layout and page

Token	What
`layout.measure.body`	body column width in `em`
`layout.table.outset`	how far tables overhang body column
`layout.figure.outset`	how far figures overhang
`layout.footnote.rule_width`	width of the rule above footnotes
`layout.sidebar.anchor_offset`	sidenote anchor displacement
`page.margin.top` / `.right` / `.bottom` / `.left`	absolute margins
`page.margin.inner` / `.outer`	mirrored margins (override .left/.right)

Component-shape tokens

Token	What
`radius.code`	code block corner radius
`code.gutter.opacity`	gutter fill opacity (0–1)
`quote.rule_width`	left rule width
`table.inline_leeway`	how much a table may exceed the body column
`list.item_gap`	
`footnote.area.height`	reserved area at page foot
`caption.gap`	gap between figure and caption
`generated.text.opacity`	

> Token values are validated at parse time. Bad enum, bad length unit, or a typo'd key all surface as `SettingsDiagnostic` errors before the renderer starts pagination.

Policy reference

Policies are typed behavioral knobs with enumerated allowed values. The registry is `policy_allowed_values()` in `crates/render-settings/src/lib.rs`.

Paragraphs

Policy	Allowed
`paragraphs.style`	`indented-prose`, `spaced-prose`
`paragraphs.first_line_indent`	`1.15em`, `1.25em`, `0pt`
`paragraphs.paragraph_gap`	`0pt`, `0.38em`, `0.65em`, `0.75em`
`paragraphs.suppress_indent_at_flow_start`	`true`, `false`

> `paragraphs.suppress_indent_after` (free-form comma list of role names) tells the renderer which preceding roles cause the next paragraph to drop its first-line indent — e.g. `chapter-heading,section-heading,scene-break,blockquote,code-block`.

Typography behavior

Policy	Allowed
`typography.justification`	`justified`, `ragged`
`typography.hyphenation`	`liang-english-us`, `none`
`typography.inline_harmony.enabled`	`true`, `false`
`typography.inline_harmony.x_height`	`true`, `false`
`typography.inline_harmony.cap_height`	`true`, `false`
`typography.inline_harmony.baseline`	`true`, `false`

> Inline harmony scales inline runs (inline code, emphasis, strong, links) so their x-height, cap-height, or baseline align with the body. The `targets` policy (free-form) lists which inline kinds participate.

Code listings

Policy	Allowed
code.wrap	soft-wrap, preserve-and-diagnose, clip-explicit
code.line_numbers	none, logical-lines, visual-lines
code.highlighting.enabled	true, false
code.highlighting.engine	syntect
code.highlighting.theme	faux-print-light
code.highlighting.unknown_language	plain
code.highlighting.preserve_source_text	true, false

Tables

Policy	Allowed
tables.sizing	intrinsic-fit, fixed, fractional, mixed
tables.snap	body-grid, none
tables.alignment	start, center, end
tables.row_breaks	between-rows-only, inside-rows-if-allowed, emergency-inside-rows
tables.header_repeat	every-fragment, after-first-fragment, none
tables.inline_leeway	50%-margin, none

Footnotes and structural

Policy	Allowed
footnotes.placement	page-local, end-of-chapter, popover
headings.keep_with_next	true, false
widows_orphans.enabled	true, false
spacing.adjacency.heading_before_suppressed_at_flow_start	true, false
spacing.adjacency.quote_series_compaction	true, false
spacing.adjacency.preserve_paragraph_rhythm	true, false

Publication-role page behavior

For each of `prefix`, `part`, `chapter`, `section`, `appendix`:

Policy	Allowed
`publication.roles.<role>.page_start`	`flow`, `new-page`, `recto`
`publication.roles.<role>.opening_treatment`	`none`, `chapter-opener`

That's how you control whether a chapter starts a new page, or only on the right-hand (recto) page, and whether it gets a drop cap / title block.

Diagnostics

Policy	Allowed
`diagnostics.visual_audit.enabled`	`true`, `false`
`diagnostics.settings_trace.enabled`	`true`, `false`

Targets

The target picks page geometry and a target-appropriate baseline. All designs honor the target's geometry unless you explicitly override `page.margin.*` and `layout.measure.body`.

Target	Page size	Use
`print-a4`	210 × 297 mm	global standard, long reports
`print-letter`	8.5 × 11 in	US standard
`print-trade-paperback`	~5.5 × 8.5 in	novels, narrative non-fiction
`mobile`	tall narrow	proof on phone
`pdf`	legacy default	unspecified PDF
`canvas`	preview	live web canvas
`epub`	reflowable	EPUB output
`web`	web	HTML/web target

Themes (the visual-taste axis)

A small set of complete palettes. Each theme sets body/heading/code font family defaults plus the core color tokens. Your design YAML can extend `intent.theme: <name>` to pick one as the starting point, then override anything.

Theme	Body family	Heading family	Code family	Vibe
`warm-literary`	Fraunces	Fraunces	(default)	warm literary print
`quiet-editorial`	Fraunces	Fraunces	(default)	quiet, longform
`crisp-technical`	Noto Serif	Noto Sans	Fira Code	clean technical
`academic-classic`	Times New Roman	Times New Roman	(default)	traditional academic
`compact-reference`	(inherits)	(inherits)	(inherits)	denser body, tighter list gaps

Policy packs (the behavior axis)

Pack	Best for	Headline policies
`book-print`	novels, narrative non-fiction	indented prose, justified + hyphenated, chapter starts new page with opener, foot-notes page-local
`report-print`	reports, essays	spaced prose, ragged + hyphenated, looser leading
`technical-print`	manuals, technical books	spaced prose, ragged, soft-wrap code, syntect highlighting
`web-readable`	web/canvas output	spaced prose, inline harmony on, no header repeat on tables
`proof-debug`	proofreading	enables visual audit + settings trace diagnostics

Component kits (the chrome axis)

Kit	Blockquote	Code	Table
`literary-minimal`	`literary-rule`	`print-soft`	`minimal-grid`
`quiet-editorial`	`editorial-rule`	`print-soft`	`quiet-grid`
`technical-lined`	`technical-callout`	`technical-lined`	`lined-grid`

Builtin design presets

These are the ready-to-go bundles. Each picks a `(target, theme, policy pack, component kit, quality)` tuple. Use as `builtin:<name>`.

Builtin	Target default	Theme	Policy	Components	Use
`book-novel`	trade paperback	warm-literary	book-print	literary-minimal	longform fiction / narrative non-fiction
`book-chapter`	trade paperback	quiet-editorial	book-print	quiet-editorial	single-chapter excerpts
`book-modern`	trade paperback	quiet-editorial	book-print	quiet-editorial	modern editorial
`academic-paper`	A4	academic-classic	report-print	quiet-editorial	papers
`technical-manual`	A4	crisp-technical	technical-print	technical-lined	manuals, software docs
`web-article`	web	quiet-editorial	web-readable	quiet-editorial	longform web
`editorial-essay`	pdf	warm-literary	report-print	quiet-editorial	essays
`editorial-feature`	pdf	warm-literary	report-print	quiet-editorial	features
`magazine-news`	pdf	warm-literary	report-print	quiet-editorial	magazine news
`slide-display`	canvas	crisp-technical	web-readable	quiet-editorial	slides
`marketing-one-page`	canvas	crisp-technical	web-readable	quiet-editorial	one-pagers
`screenplay`	pdf	compact-reference	report-print	quiet-editorial	scripts
`note-personal`	pdf	compact-reference	report-print	quiet-editorial	personal notes

Fonts

Fonts resolve dynamically through Google Fonts. Use the family name exactly as it appears on the Google Fonts site:

```
tokens:
  typography.body.family: Fraunces
  typography.heading.family: DM Sans
  typography.code.family: Fira Code
```

Lookup order:

1. **System fonts** — anything installed in `~/Library/Fonts` or system font directories. Pass the family name as the system reports it.
2. **Local cache** — `~/.cache/faux-press/fonts/<identifier>.ttf`.
3. **Google Fonts via jsdelivr CDN** — first run hits the network, writes the TTF into the cache, subsequent runs hit disk.

Variable-axis fonts (Fraunces, Inter, Source Serif 4, etc.) ship as a single TTF and cover all weights/styles. You don't need to install per- style files for modern Google Fonts.

Bring `faux-press doctor` if you suspect a font isn't resolving.

Building your own design — recipes

Recipe 1: minimal override on top of book-novel

```
schema: faux.design/v1
id: my.larger-margins
version: 0.1.0
name: Larger Margins Book
extends: [builtin:book-novel]
tokens:
  page.margin.left: 140pt
  page.margin.right: 80pt
  typography.body.size: 11.0pt
```

Recipe 2: dark theme for screen reading

```
schema: faux.design/v1
id: my.dark-reading
version: 0.1.0
name: Dark Reading
extends: [builtin:book-novel]
tokens:
  color.page.background: "#101820"
  color.text: "#eee8d5"
  color.heading.text: "#fff4c2"
  color.initial_letter.text: "#f7b267"
  color.footnote.text: "#d7d0c7"
  color.code.background: "#0d141d"
  color.code.text: "#e9e3d4"
  color.code.gutter: "#0a1018"
  color.code.line_number: "#66615a"
  color.code.keyword: "#ffb86c"
  color.code.string: "#a7c080"
  color.code.comment: "#8b949e"
  color.code.literal: "#c792ea"
  color.table.cell_background: "#111827"
  color.table.alt_row_background: "#182033"
  color.table.hairline: "#4b5563"
policies:
  publication.roles.chapter.page_start: flow
  typography.justification: ragged
  typography.hyphenation: none
```

Recipe 3: technical manual with ragged right + soft-wrap code

```
schema: faux.design/v1
id: my.manual
version: 0.1.0
name: My Technical Manual
extends: [builtin:technical-manual]
tokens:
  typography.body.family: Source Serif 4
  typography.heading.family: Inter
  typography.code.family: JetBrains Mono
  typography.body.size: 10.8pt
  typography.body.leading: 1.48
policies:
  paragraphs.style: spaced-prose
  paragraphs.paragraph_gap: 0.65em
  typography.justification: ragged
  typography.hyphenation: liang-english-us
  code.line_numbers: visual-lines
  tables.header_repeat: every-fragment
```

Recipe 4: target-specific overrides

Designs can carry per-target overrides (small screens vs print) so a single design covers both:

```
schema: faux.design/v1
id: my.book
extends: [builtin:book-novel]
tokens:
  typography.body.size: 11.4pt
targets:
  - target: mobile
    tokens:
      typography.body.size: 16pt
      typography.body.leading: 1.55
  - target: print-a4
    tokens:
      page.margin.left: 138pt
      page.margin.right: 76pt
```

Where things live

What	Path
Token registry	`crates/render-settings/src/lib.rs :: token_registry`
Policy allowed values	`crates/render-settings/src/lib.rs :: policy_allowed_values`
Theme defaults	`crates/render-settings/src/lib.rs :: apply_theme_defaults`
Policy-pack defaults	`crates/render-settings/src/lib.rs :: apply_policy_pack_defaults`
Component-kit defaults	`crates/render-settings/src/lib.rs :: apply_component_kit_defaults`
Builtin preset bundles	`crates/render-settings/src/lib.rs :: preset_alias_settings`
Google Fonts fetch	`crates/font-resolver/`
Publication roles	`crates/atlas-types/src/publication.rs`
Sample book	`books/the-honest-machine/`
Sample with 12 designs	`~/Downloads/Projects/reliability-book/themes/`

Debugging cascade

```
faux-press explain typography.body.family chapter.md
```

...alks the cascade and prints every layer that touched the value and which layer won. `faux-press inspect` prints the publication tree (so you can see what got identified as a chapter, section, scene break, etc.). `faux-press lint` runs static typographic checks. `faux-press proof` emits per-decision artifacts for any rendered output.

When in doubt: render with `--trace` and tail the trace log; every solver, placer, and paint decision is keyed by the same identity that appears in the PDF metadata.

Summary

You want to …	Edit this
change which fonts are used	`typography.body.family`, `typography.heading.family`, `typography.code.family`
make the book dark	`color.page.background` + the `color.*` family
pick a different page size	`--target <name>` (or `[output.faux] target`)
start chapters on the right page only	`publication.roles.chapter.page_start: recto`
turn off justification	`typography.justification: ragged`
turn off hyphenation	`typography.hyphenation: none`
add space between paragraphs	`paragraphs.style: spaced-prose` + `spacing.paragraph.gap`
make tables span beyond the body column	`layout.table.outset`, `tables.inline_leeway`
move footnotes to end of chapter	`footnotes.placement: end-of-chapter`
change a chapter opener treatment	`publication.roles.chapter.opening_treatment: chapter-opener` / `none`
pick a totally different look	extend a different `builtin:<name>`

Everything else is a token or a policy. Look them up in the registry. Override them in your design YAML. Render.