Minimal + Custom Hooks

Combining content focus with bespoke roadmaps

Presentate Team 2026-01-23

Part 0

Theme Synergy

Theme Synergy

OUTLINE

The Best of Both Worlds

How Hooks Overlap

SECTION STRUCTURE

Theme Synergy

OUTLINE

The Best of Both Worlds

How Hooks Overlap

SECTION STRUCTURE

Clean Canvas
Bespoke Transitions

0 Clean Canvas

This example uses the `minimal` theme, which provides a clean canvas without persistent UI elements (no sidebars, headers, or footers).

0 Bespoke Transitions

We've injected the complex transition slides from the `custom-transition` example using **Hooks**.

Theme Synergy

OUTLINE

The Best of Both Worlds

How Hooks Overlap

SECTION STRUCTURE

Theme Synergy

OUTLINE

The Best of Both Worlds

How Hooks Overlap

SECTION STRUCTURE

Default Engine
Manual Override

0 Default Engine

The `minimal` theme would normally use the **Unified Transition Engine** to show a simple roadmap.

0 Manual Override

By providing `on-section-change` and `on-subsection-change` functions, you override the engine's default behavior with your own logic.

Part 0

Configuration Details

Configuration Details

OUTLINE

Injecting the Logic

Numbering Propagation

Source Code: Section Hook

Source Code: Subsection Hook

SECTION STRUCTURE

Configuration Details

OUTLINE

Injecting the Logic

Numbering Propagation

Source Code: Section Hook

Source Code: Subsection Hook

SECTION STRUCTURE

0 Injecting the Logic

The injection is done via the `template` parameters:

```
#show: template.with(
  on-part-change: my-section-transition,
  on-section-change: my-subsection-transition,
  ...
)
```

Configuration Details

OUTLINE

Injecting the Logic

Numbering Propagation

Source Code: Section Hook

Source Code: Subsection Hook

SECTION STRUCTURE

Configuration Details

OUTLINE

Injecting the Logic

Numbering Propagation

Source Code: Section Hook

Source Code: Subsection Hook

SECTION STRUCTURE

0 Numbering Propagation

Even with custom hooks, the global `show-heading-numbering` and `numbering-format` options are respected.

Configuration Details

OUTLINE

Injecting the Logic

Numbering Propagation

Source Code: Section Hook

Source Code: Subsection Hook

SECTION STRUCTURE

Configuration Details

OUTLINE

Injecting the Logic

Numbering Propagation

Source Code: Section Hook

Source Code: Subsection Hook

SECTION STRUCTURE

0 Source Code: Section Hook

Used for `on-part-change` (Level 1) in this mapping:

```
#let my-section-transition(h) = empty-slide(fill: eastern, {
  set text(fill: white)
  set align(center + horizon)
  let part-num = counter(heading).at(h.location()).at(0)
  text(size: 1.2em, white.transparentize(30%), smallcaps[Part #part-num])
  v(0.5em)
  text(size: 1.8em, weight: "bold", h.body)
  v(1em)
  line(length: 40%, stroke: 0.5pt + white)
})
```

Configuration Details

OUTLINE

Injecting the Logic

Numbering Propagation

Source Code: Section Hook

Source Code: Subsection Hook

SECTION STRUCTURE

Configuration Details

OUTLINE

Injecting the Logic

Numbering Propagation

Source Code: Section Hook

Source Code: Subsection Hook

SECTION STRUCTURE

0 Source Code: Subsection Hook

Used for `on-section-change` (Level 2) in this mapping:

```
#let my-subsection-transition(h) = {
  let active = get-active-headings(h.location())
  let is-first = counter(heading).at(h.location()).at(1, default: 1) == 1

  empty-slide({
    // ... Part title logic ...
    grid(columns: (1fr, 1fr),
      // Left: Current Section Highlighting
      progressive-outline(
        level-1-mode: "none", level-2-mode: "current-parent",
        target-location: if not (is-first and sub == 1) { h.location() } else { active.h1.location() },
        // ... styles ...
      ),
      // Right: Future Subsection preview
      uncover(if is-first { 2 } else { 1 })[
        progressive-outline(
          level-1-mode: "none", level-2-mode: "none", level-3-mode: "current-parent",
          target-location: h.location(),
          // ... styles ...
        )
      ]
    )
  })
}
```

Part 0

Conclusion

Conclusion

OUTLINE

SECTION STRUCTURE

Summary

Conclusion

OUTLINE

SECTION STRUCTURE

Summary

0 Summary

The hook system provides maximum flexibility:

- Use `minimal` for content focus.
- Use custom functions for high-impact transitions.
- Maintain structural consistency via the global configuration.