Formatting & Branding

Session 5

Mickaël CANOUIL, Ph.D.[1*]

Saturday, the 7th of February, 2026

Affiliations

[1] Independent Contractor, Lille, France

ORCID

Mickaël CANOUIL, Ph.D.: 0000-0002-3396-4549

[*]Corresponding author: pro@mickael.canouil.dev

Table of Contents

Table of Contents

Table of Contents

1. Session Overview

1.1. Session 5: Formatting & Branding

- Bootstrap Theming & Customisation
  - Bootstrap 5 integration and built-in theme selection.
  - Progressive customisation from YAML options to light CSS.
  - Theme layering and CSS custom properties.
- Unified Branding with Brand.yml
  - Brand.yml systems for cross-format consistency.
  - Colour palettes, typography, and logo management.
  - Integration with Bootstrap and format-specific options.
- Pandoc Templating & Extensions
  - Understanding Pandoc's templating syntax fundamentals.
  - Variables, conditionals, and loops for dynamic content.
  - Developing brand extensions for organisational reuse.
- Typst Customisation
  - Typst templates and partials for PDF branding.
  - Template structure and brand.yml integration.
  - Creating custom Typst formats with consistent visual identity.

1.2. Session Objectives

This session explores Quarto's advanced theming and branding capabilities, covering Bootstrap 5 customisation, `brand.yml` systems for unified branding, Pandoc templating syntax, and Typst customisation for professional PDF output.

1.3. Learning Objectives

By the end of this session, participants will be able to:

- Implement professional themes with Bootstrap 5 customisation.
- Apply progressive customisation from YAML to light CSS.
- Create unified branding across multiple output formats using `brand.yml`.
- Develop brand extensions for organisational reuse.
- Understand Pandoc's templating syntax for dynamic content generation.
- Customise Typst templates and partials for branded PDF output.

2. Theming (Bootstrap & Customisation)

2.1. Bootstrap in Quarto

Quarto HTML format uses Bootstrap 5, providing:

- Several built-in themes from Bootswatch.
- Responsive grid system.
- Component library (e.g., buttons, cards, navigation).
- Customisation system via CSS variables.

2.2. Theme Selection

Available themes: default, cerulean, cosmo, cyborg, darkly, flatly, journal, litera, lumen, lux, materia, minty, morph, pulse, quartz, sandstone, simplex, sketchy, slate, solar, spacelab, superhero, united, vapor, yeti, zephyr.

See HTML Theming for more.

- Single theme.

```yaml
format:
  html:
    theme: cosmo
```

- Light/dark theme pair.

```yaml
format:
  html:
    theme:
      light: cosmo
      dark: darkly
```

- Theme + customisation.

```yaml
format:
  html:
    theme: [cosmo, custom.scss]
```

Can also be written as:

```yaml
format:
  html:
    theme:
```

```
    - cosmo
    - custom.scss
```

2.3. Basic Customisation

```yaml
format:
  html:
    theme: cosmo

    # Typography
    fontsize: 1.1em
    linestretch: 1.6
    mainfont: "Inter"
```

```yaml
format:
  html:
    theme: cosmo

    # Typography
    fontsize: 1.1em
    linestretch: 1.6
    mainfont: "Inter"

    # Colours
    linkcolor: "#2E86AB"
    backgroundcolor: "#FFFFFF"
```

```yaml
format:
  html:
    theme: cosmo

    # Typography
    fontsize: 1.1em
    linestretch: 1.6
    mainfont: "Inter"

    # Colours
    linkcolor: "#2E86AB"
    backgroundcolor: "#FFFFFF"

    # Layout
    max-width: 1200px
```

```
    margin-left: 2rem
    margin-right: 2rem
```

See HTML Theming - Basic Options for more.

2.4. Progressive Enhancement

Move to SCSS for more control:

YAML

```yaml
format:
  html:
    theme:
      - cosmo
      - styles.scss
```

Custom SCSS file follows a specific structure using comments:

styles.scss

```scss
/*-- scss:defaults --*/
$h2-font-size:          1.6rem !default;
$headings-font-weight:  500 !default;

/*-- scss:rules --*/
h1, h2, h3, h4, h5, h6 {
  text-shadow: -1px -1px 0 rgba(0, 0, 0, .3);
}
```

See More About Quarto Themes for more.

3. `brand.yml` Foundation

3.1. Understanding Brand.yml

`brand.yml` is a single YAML file that codifies your organisation's brand guidelines into a format that can be used by Quarto (but not limited to) to create branded outputs across multiple formats.

Key benefits:

- Cross-format consistency - HTML, PDF, presentations share the same design.
- Organisational standards - Teams work with unified branding.
- Easy maintenance - Update once, apply everywhere.
- Simple syntax - No complex CSS knowledge required.

3.2. Basic `brand.yml` Structure

Create a `_brand.yml` file in your project root:

_brand.yml

```yaml
meta:
  name: "Your Organisation"
  link: "https://yourorg.com"                    (1)

color:
  palette:
    primary: "#2E86AB"
    secondary: "#A23B72"
    light: "#F8F9FA"
    dark: "#343A40"
  primary: primary
  foreground: dark
  background: light                              (2)

logo:
  small: "assets/logo-small.png"
  medium: "assets/logo-medium.png"
  large: "assets/logo-large.png"                 (3)

typography:
  fonts:
    - family: "Inter"
      source: "google"
  base: "Inter"
  headings: "Inter"                              (4)
```

(1) Basic brand identity.
(2) Colour palette system.
(3) Logo management.
(4) Typography system.

3.3. Colour Palette System

The palette defines named colours that can be referenced throughout your brand:

_brand.yml

```yaml
color:
  palette:
    ocean-blue: "#2E86AB"
    coral-pink: "#A23B72"
    sunshine: "#F4A261"
    forest: "#2A9D8F"
    charcoal: "#264653"
    pearl: "#F8F9FA"                    (1)
  # Semantic assignments
  primary: ocean-blue
  secondary: coral-pink
  success: forest
  warning: sunshine
  foreground: charcoal
  background: pearl
```

(1) Define a palette of named colours.

Cross-format usage:

- HTML: Available as `$brand-primary` (SCSS), `--brand-primary` (CSS), etc.
- Typst: Available as `brand-color.primary`, etc.

3.4. Typography Configuration

Define your organisation's font system:

YAML

```yaml
typography:
  fonts:
    # Google Fonts (automatically loaded)
    - family: "Inter"
      source: "google"
      weight: [300, 400, 500, 600, 700]

    # System fonts (fallback)
    - family: "system-ui"
      source: "system"

  # Font assignments
  base: "Inter"           # Body text
  headings: "Inter"       # All headings
  monospace: "SF Mono"    # Code blocks
```

See Typst Brand YAML for troubleshooting fonts in Typst.

3.5. Logo Management

Provide logos at different sizes for various contexts:

```yaml
logo:
  # Favicon and small contexts
  small: "brand/favicon.png"

  # Navigation bars, headers
  medium: "brand/logo-nav.png"

  # Hero sections, print materials
  large: "brand/logo-full.png"
```

Format	Location	Logo Preference (high to low)
`html` / `dashboard`	Top navigation bar	`small` > `medium` > `large`
`html`	Side navigation	`medium` > `small` > `large`
`typst`	Top left, control with `format: typst: logo`	`medium` > `small` > `large`
`revealjs`	Bottom right corner of slides	`medium` > `small` > `large`
`website` / `book` project	`favicon` shown in browser tab	`small`

See Logo Configuration for more.

Format	Location	Logo Preference (high to low)
`html` / `dashboard`	Top navigation bar	`small` > `medium` > `large`
`html`	Side navigation	`medium` > `small` > `large`
`typst`	Top left, control with `format: typst: logo`	`medium` > `small` > `large`
`revealjs`	Bottom right corner of slides	`medium` > `small` > `large`
`website` / `book` project	`favicon` shown in browser tab	`small`

See Logo Configuration for more.

3.6. Brand Integration

Combine `brand.yml` with Bootstrap customisation:

_quarto.yml

```yaml
format:
  html:
    theme:
      - cosmo
      - brand
      - styles.scss
    css: custom-components.css
```

(1)

(1) Prefer `brand.yml` or SCSS for component customisation.

Note

The brand colours become available as CSS variables and SCSS variables automatically.

4. Brand Extensions Development

4.1. What Are Brand Extensions?

Brand extensions are Quarto extensions that provide a `brand.yml` file and its assets, packaged for easy distribution and reuse.

Key features:

- Portable branding - Install everywhere.
- Team consistency - Everyone gets the same brand.
- Asset management - Logos and fonts included.
- Version control - Track brand changes over time.

4.2. Creating a Brand Extension

4.2.1. Step 1: Scaffold Extension

```bash
quarto create extension brand my-brand
cd my-brand
```

This creates:

```
my-brand
├── README.md
├── _extensions
│   `-- my-brand
│       ├── _extension.yml
│       `-- brand.yml
├── _quarto.yml
`-- example.qmd

2 directories, 5 files
```

4.2.2. Step 2: Configure Extension

```yaml
title: My-brand
author: First Last
version: 1.0.0
quarto-required: "≥1.9.0"
contributes:
  metadata:
    project:
      brand: brand.yml
```

4.2.3. Step 3: Design Brand System

_extensions/my-brand/brand.yml

```yaml
meta:
  name: "My Organisation"
  link: "https://myorg.com"

color:
  palette:
    # Primary brand colours
    brand-blue: "#2E86AB"
    brand-coral: "#A23B72"
    brand-gold: "#F4A261"
    # Supporting colours
    neutral-light: "#F8F9FA"
    neutral-dark: "#343A40"
    success-green: "#28A745"
    warning-amber: "#FFC107"
  # Semantic mappings
  primary: brand-blue
  secondary: brand-coral
  success: success-green
  warning: warning-amber
  foreground: neutral-dark
  background: neutral-light

logo:
  small: assets/logo/favicon.png
  medium: assets/logo/logo-
horizontal.png
  large: assets/logo/logo-
vertical.png

typography:
  fonts:
    - family: "Source Sans Pro"
      source: "google"
      weight: [300, 400, 600, 700]

  base: "Source Sans Pro"
```

_extensions/my-brand/brand.yml

```yaml
defaults:
  bootstrap:
    defaults:
      # Enhanced navigation
      navbar-padding-y: 1rem
      navbar-brand-font-size:
1.5rem

      # Improved buttons
      btn-padding-y: 0.5rem
      btn-padding-x: 1.5rem
      btn-font-weight: 600

      # Better spacing
      spacer: 1rem

    rules: |
      /* Professional enhancements
*/
      .navbar-brand {
        font-weight: 700;
      }

      .btn-primary {
        border-radius: 2rem;
        transition: all 0.3s ease;
      }

      .btn-primary:hover {
        transform:
translateY(-1px);
      }

      /* Content styling */
      .title-block-header {
        padding: 3rem 2rem;
        margin-bottom: 2rem;
        border-radius: 0.5rem;
      }
```

4.2.4. Step 4: Add Assets

Organise brand assets:


```txt
_extensions/my-brand/
├── _extension.yml
├── brand.yml
└── assets/
    ├── logo/
    │   ├── favicon.png
    │   ├── logo-horizontal.png
    │   └── logo-vertical.png
    └── fonts/
        └── custom-font.woff2
```

4.2.5. Step 5: Create Example

example.qmd

```yaml
---
title: "Brand Extension Showcase"
subtitle: "Demonstrating consistent branding"
author: "Your Name"
format:
  html:
    theme:
      - quartz
      - brand
---

# Welcome to Our Brand System

This document demonstrates the power of brand extensions in Quarto.

[Primary Action]{.btn .btn-primary}
[Secondary Action]{.btn .btn-secondary}
```

! Important

Brand extensions require a `_quarto.yml` project file to work.

5. Hands-On Exercise: Building Your Brand Extension

5.1. Exercise Overview

Objective: Create a comprehensive brand extension that demonstrates unified branding across formats using `brand.yml` and light CSS customisation.

Example Code: Exercises Brand

```bash
tar -xzf "05a-exercises.tar.gz" -C "05-formatting-branding"
```

5.2. Part 1: Brand Extension Setup

1. Generate extension scaffold:

```bash
quarto create extension brand my-company-brand
cd my-company-brand
```

2. Define your brand identity.
3. Add styling.

5.3. Part 2: Testing

1. Build test project:

```bash
quarto create project
```

2. Test across formats:

```bash
# Test HTML output
quarto render example.qmd --to html

# Test presentation
quarto render example.qmd --to revealjs

# Test Typst output
quarto render example.qmd --to typst
```

3. Verify brand consistency.

5.4. Success Criteria

☑ You've successfully completed the exercise if you can:

- Create a working brand extension with consistent visual identity.
- Implement styling using `brand.yml` and light (S)CSS.
- Test the brand system across multiple formats.
- Demonstrate cross-format consistency.
- Build a reusable brand asset for teams.
- Apply design principles effectively.

6. Pandoc Templating System

6.1. Understanding Pandoc Templates

Quarto uses Pandoc's templating system to generate output.

Key concepts:

- Variables - `$variable$` inserts YAML metadata.
- Conditionals - `$if(variable)$ ... $endif$` for optional content.
- Loops - `$for(variable)$ ... $endfor$` for repeated content.
- Partials - Reusable template components.

See Pandoc Template Syntax for complete documentation.

6.2. Basic Variable Syntax

Variables from YAML front matter are inserted using `$variable$`:

Quarto Document

```yaml
---
title: "My Report"
author: "Jane Smith"
date: "2025-01-15"
---
```

Template Input

```
Title: $title$
Author: $author$
Date: $date$
```

Template Output

```
Title: My Report
Author: Jane Smith
Date: 2025-01-15
```

> **Tip**
>
> Use `$variable$` in any template file (`.typ`, `.tex`, `.html`, etc.).

See Article Templates for templates and partials.

6.3. Conditional Content

Show content only when a variable exists:

Quarto Document

```yaml
---
title: "Report"
subtitle: "Q4 Analysis"
# No author specified
---
```

Template Input

```
Title: $title$

$if(subtitle)$
Subtitle: $subtitle$
$endif$

$if(author)$
Author: $author$
$endif$
```

Note

Use `$if(variable)$ ... $else$ ... $endif$` for alternative content.

Template Output

```
Title: Report
Subtitle: Q4 Analysis
```

Important

`$if(variable)$` in Pandoc templates checks if a variable is:

- Defined (exists in the metadata).
- Non-empty (has a value).
- Not explicitly false.

6.4. Loops for Multiple Items

Process lists from YAML metadata:

Quarto Document

```yaml
---
authors:
  - name: Jane Smith
    email: jane@example.com
  - name: John Doe
    email: john@example.com
---
```

Template Input

```
Authors:
$for(authors)$
- $it.name$ ($it.email$)
$endfor$
```

Template Output

```
Authors:
- Jane Smith (jane@example.com)
- John Doe (john@example.com)
```


> **Note**
>
> Use `$it$` to reference the current item in a loop. Use `$sep$` followed by the separator string for separators.

6.5. Separator Usage

Add separators between items:

Quarto Document

```yaml
---
tags:
  - Research
  - Data Science
  - Machine Learning
---
```

Template Input

```
Tags: $for(tags)$$it$$sep$,
$endfor$
```

Template Output

```
Tags: Research, Data Science,
Machine Learning
```

Template Input

```
Tags: $for(tags)$$it$$endfor$
```

Template Output

```
Tags: ResearchData ScienceMachine
Learning
```


7. Hands-On Exercise: Pandoc Templating

7.1. Exercise Overview

Objective: Modify a partial template to display additional metadata using Pandoc templating syntax.

Example Code: Exercises Partials

```bash
tar -xzf "05b-exercises.tar.gz" -C "05-custom-partials"
```

7.2. Part 1: Template Modification

1. Retrieve the partials described in Article Templates
   - Typst Partials
   - HTML Partials
   - Reveal.js Partials
2. Modify the title slide partial from Reveal.js to underline the corresponding author:
   - Use `$if(by-author)$` to check for `by-author` metadata.
   - Loop through `by-author` to find the corresponding author and underline their name.
   - See Author Schema for details.

7.3. Part 2: Customise Further

1. Add additional metadata to the title slide, such as affiliations or ORCID IDs.
2. Test the changes by rendering a sample presentation with multiple authors with affiliations and a corresponding author.

7.4. Success Criteria

☑ You've successfully completed the exercise if you can:

- Modify a Quarto/Pandoc partial template using conditionals and loops.
- Display dynamic content based on YAML metadata.
- Test and verify the output in a Reveal.js presentation.

8. Typst Customisation (For the Curious)

8.1. Typst in Quarto

Typst is a modern typesetting system for creating beautiful PDFs, designed to be simpler than LaTeX.

Why Typst?

- Fast - Instant PDF generation.
- Simple - Markdown-like syntax.
- Flexible - Easy template customisation.

See Typst Basics for more.

8.2. Typst Template Structure

Quarto's Typst templates use two key files:

- `typst-template.typ` - Defines document layout:

typst-template.typ

```
#let article(title: none, authors: none, body) = {
  set page(paper: "a4", margin: 2.5cm)                    (1)

  set text(font: "Inter", size: 11pt)                     (2)

  if title ≠ none {
    align(center, text(weight: "bold", 1.5em, title))
  }                                                       (3)

  body                                                    (4)
}
```

(1) Page setup.
(2) Typography settings.
(3) Conditional title block.
(4) Main content insertion.

- `typst-show.typ` - Captures YAML metadata using Pandoc syntax:

typst-show.typ

```
#show: doc ⇒ article(
  $if(title)$
  title: [$title$],
  $endif$
  $if(by-author)$
  authors: [$for(by-author)$$it.name.literal$$sep$, $endfor$],
  $endif$                                                 (1)
  doc,
)
```


① Use `by-author` to get Quarto's extended author metadata.

> **Note**
>
> Notice the Pandoc template syntax: `$if()$`, `$for()$`, `$variable$`.

8.3. Simple Customisation Example

typst-template.typ

```
#let article(
  title: none,
  body
) = {
  let primary = rgb("#2E86AB")

  if title ≠ none {
    block(
      fill: primary,
      inset: 1em,
      radius: 0.5em,
      width: 100%
    )[
      #align(center)[
        #text(
          fill: white,
          weight: "bold",
          size: 1.5em,
          title
        )
      ]
    ]
  }

  body
}
```

typst-show.typ

```
#show: doc ⇒ article(
  $if(title)$
  title: [$title$],
  $endif$
  doc,
)
```

Figure 8.1: Screenshot of a PDF document with blue header titled Demo of Typst Format on white background with page number 1 at bottom

8.4. Creating a Typst Format

8.4.1. Step 1: Scaffold Extension

```bash
quarto create extension format:typst my-format
cd my-format
```

This creates:

```
my-format
├── README.md
├── _extensions
│   `-- my-format
│       ├── _extension.yml
│       ├── typst-show.typ
│       `-- typst-template.typ
`-- template.qmd

2 directories, 5 files
```

> **Tip**
>
> Modify the generated `typst-template.typ` file to customise your layout.

8.4.2. Step 2: Integration with `brand.yml`

_brand.yml

```yaml
color:
  primary: "#2E86AB"

logo:
  medium: "assets/logo.png"
```

typst-show.typ

```
#show: doc => article(
  $if(title)$
  title: [$title$],
  title_colour: brand-
color.primary,
  $endif$
  doc,
)
```

page.typ

```
$if(logo)$
#set page(
  header: context {
    if counter(page).get().first()
== 1{
      place(top + left, dy: 0.5cm,
image("$logo.path$", width: 2cm))
    }
  }
)
$endif$
```

typst-template.typ

```
#let article(
  title: none,
  title_colour: "#000000",
  body
) = {
  set text(size: 11pt)
  if title ≠ none {
    align(center)[
      #text(
        fill: rgb(title_colour),
        weight: "bold",
        size: 1.5em,
        title
      )
    ]
  }
  body
}
```

9. Hands-On Exercise: Creating a Format Extension

9.1. Exercise Overview

Objective: Create a custom format extension that bundles `brand.yml` and styling into a reusable format for your organisation.

Example Code: Exercises Typst

```bash
tar -xzf "05c-exercises.tar.gz" -C "05-format-extension"
```

9.2. Part 1: Format Extension Setup

1. Generate format extension scaffold:

```bash
quarto create extension format:html company-report
cd company-report
```

2. Add brand.yml and configure extension in _extension.yml`:

```yaml
contributes:
  formats:
    typst:
      template-partials:
        - assets-typst/typst-template.typ
        - assets-typst/typst-show.typ
        - assets-typst/page.typ
  metadata:
    project:
      brand: brand.yml
```

9.3. Part 2: Customise and Test Your Format

1. Define your template in `typst-template.typ` and `typst-show.typ`.
2. Use your format in `template.qmd`:

```yaml
---
title: "Quarterly Report"
format: company-report-typst
---
```

3. Render and verify:

```bash
quarto render template.qmd
```

9.4. Part 3: Optional HTML Variant

Add HTML output to your format:

1. Add HTML format to `_extension.yml`.
2. Create/Modify simple template. (See HTML partials)
3. Test both formats:

```bash
quarto render template.qmd --to company-report-html
quarto render template.qmd --to company-report-typst
```

9.5. Success Criteria

☑ You've successfully completed the exercise if you can:

- Create a working format extension with bundled branding.
- Apply custom styling through brand.yml and SCSS.
- Test the format with sample documents.
- (Optional) Add HTML variant.