

Formatting & Branding

Session 5

Mickaël CANOUIL, Ph.D.^{1*}

Saturday, the 7th of February, 2026

Affiliations

¹Independent Contractor, Lille, France

ORCID

 Mickaël CANOUIL, Ph.D.: [0000-0002-3396-4549](https://orcid.org/0000-0002-3396-4549)

*Corresponding author: pro@mickael.canouil.dev

Table of Contents

- 1. Session Overview 4
 - 1.1. Session 5: Formatting & Branding 5
 - 1.2. Session Objectives 5
 - 1.3. Learning Objectives 5
- 2. Theming (Bootstrap & Customisation) 6
 - 2.1. Bootstrap in Quarto 7
 - 2.2. Theme Selection 7
 - 2.3. Basic Customisation 8
 - 2.4. Progressive Enhancement 9
- 3. brand.yml Foundation 10
 - 3.1. Understanding Brand.yml 11
 - 3.2. Basic brand.yml Structure 11
 - 3.3. Colour Palette System 12
 - 3.4. Typography Configuration 12
 - 3.5. Logo Management 13
 - 3.6. Brand Integration 13
- 4. Brand Extensions Development 15
 - 4.1. What Are Brand Extensions? 16
 - 4.2. Creating a Brand Extension 16
 - 4.2.1. Step 1: Scaffold Extension 16
 - 4.2.2. Step 2: Configure Extension 16
 - 4.2.3. Step 3: Design Brand System 17
 - 4.2.4. Step 4: Add Assets 17
 - 4.2.5. Step 5: Create Example 18
- 5. Hands-On Exercise: Building Your Brand Extension 19
 - 5.1. Exercise Overview 20
 - 5.2. Part 1: Brand Extension Setup 20
 - 5.3. Part 2: Testing 20
 - 5.4. Success Criteria 21
- 6. Pandoc Templating System 22
 - 6.1. Understanding Pandoc Templates 23
 - 6.2. Basic Variable Syntax 23
 - 6.3. Conditional Content 23
 - 6.4. Loops for Multiple Items 24
 - 6.5. Separator Usage 25
- 7. Hands-On Exercise: Pandoc Templating 26
 - 7.1. Exercise Overview 27
 - 7.2. Part 1: Template Modification 27
 - 7.3. Part 2: Customise Further 27
 - 7.4. Success Criteria 27

Table of Contents

- 8. Typst Customisation (For the Curious) 28
 - 8.1. Typst in Quarto 29
 - 8.2. Typst Template Structure 29
 - 8.3. Simple Customisation Example 30
 - 8.4. Creating a Typst Format 31
 - 8.4.1. Step 1: Scaffold Extension 31
 - 8.4.2. Step 2: Integration with `brand.yml` 32
- 9. Hands-On Exercise: Creating a Format Extension 33
 - 9.1. Exercise Overview 34
 - 9.2. Part 1: Format Extension Setup 34
 - 9.3. Part 2: Customise and Test Your Format 34
 - 9.4. Part 3: Optional HTML Variant 35
 - 9.5. Success Criteria 35

1. Session Overview

- 1.1. Session 5: Formatting & Branding
- 1.2. Session Objectives
- 1.3. Learning Objectives

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
- 2.2. Theme Selection
- 2.3. Basic Customisation
- 2.4. Progressive Enhancement

2.1. Bootstrap in Quarto

Quarto HTML format uses Bootstrap 5, providing:

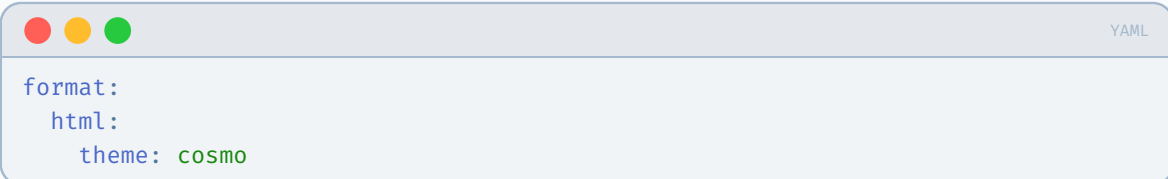
- Several built-in themes from Bootswatch.
- Responsive grid system.
- Component library (e.g., buttons, cards, navigation).
- Customisation system via CSS variables.

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.

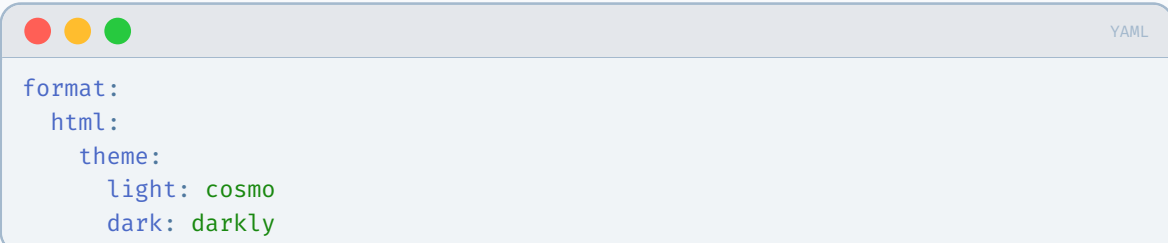
- Single theme.



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

A code block with a light blue background and a grey header containing three colored circles (red, yellow, green) and the text 'YAML'.

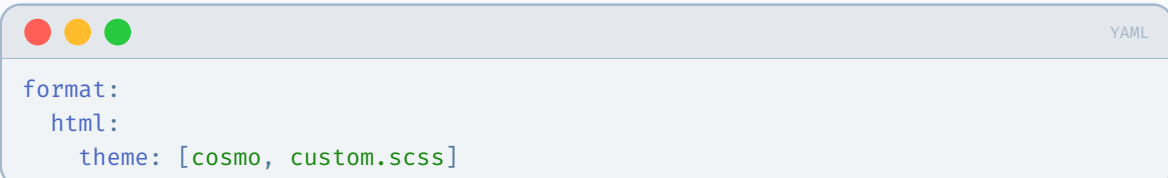
- Light/dark theme pair.



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

A code block with a light blue background and a grey header containing three colored circles (red, yellow, green) and the text 'YAML'.

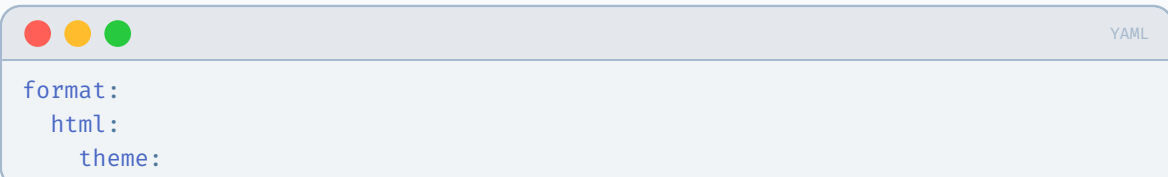
- Theme + customisation.



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

A code block with a light blue background and a grey header containing three colored circles (red, yellow, green) and the text 'YAML'.

Can also be written as:



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

A code block with a light blue background and a grey header containing three colored circles (red, yellow, green) and the text 'YAML'.

- cosmo
- custom.scss

2.3. Basic Customisation

```
format:
  html:
    theme: cosmo

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

```
format:
  html:
    theme: cosmo

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

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

```
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:

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

Custom SCSS file follows a specific structure using comments:

```
/*-- 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
- 3.2. Basic `brand.yml` Structure
- 3.3. Colour Palette System
- 3.4. Typography Configuration
- 3.5. Logo Management
- 3.6. Brand Integration

3.1. Understanding Brand.yml

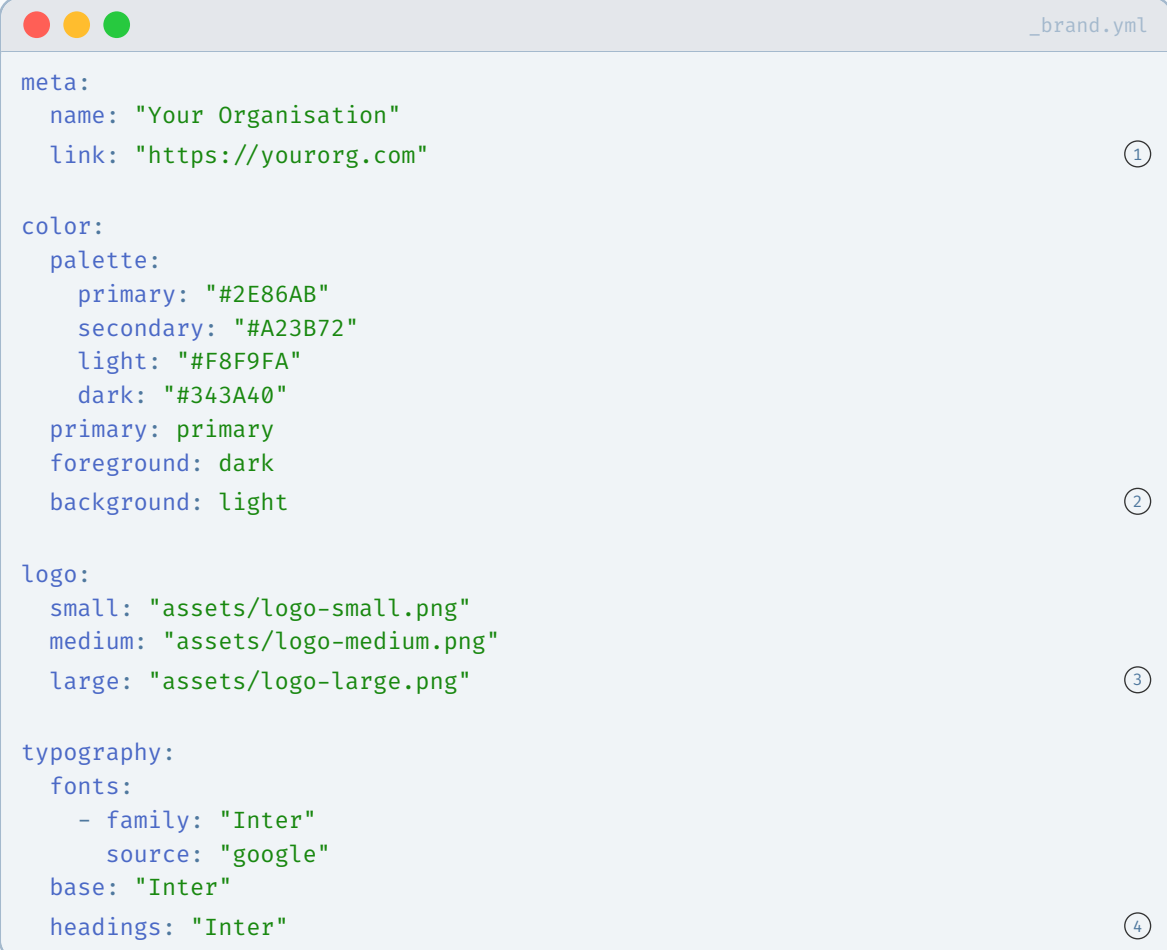
`brand.yml` is a single YAML file that codifies your organisation's brand guidelines into a format that can be used by Quarto (but not limited to) to create branded outputs across multiple formats.

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:



```
meta:
  name: "Your Organisation"
  link: "https://yourorg.com"

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

logo:
  small: "assets/logo-small.png"
  medium: "assets/logo-medium.png"
  large: "assets/logo-large.png"

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

- 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

color:
  palette:
    ocean-blue: "#2E86AB"
    coral-pink: "#A23B72"
    sunshine: "#F4A261"
    forest: "#2A9D8F"
    charcoal: "#264653"
    pearl: "#F8F9FA"
  # 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

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

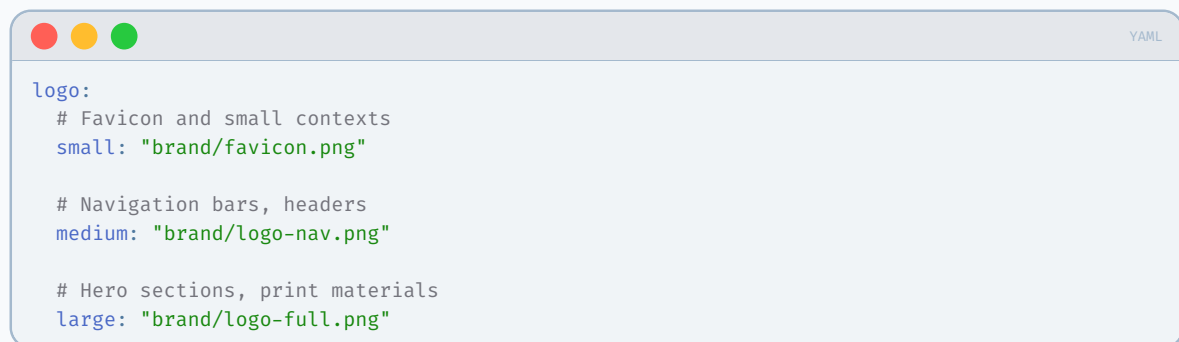
    # System fonts (fallback)
    - family: "system-ui"
      source: "system"

  # 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:



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

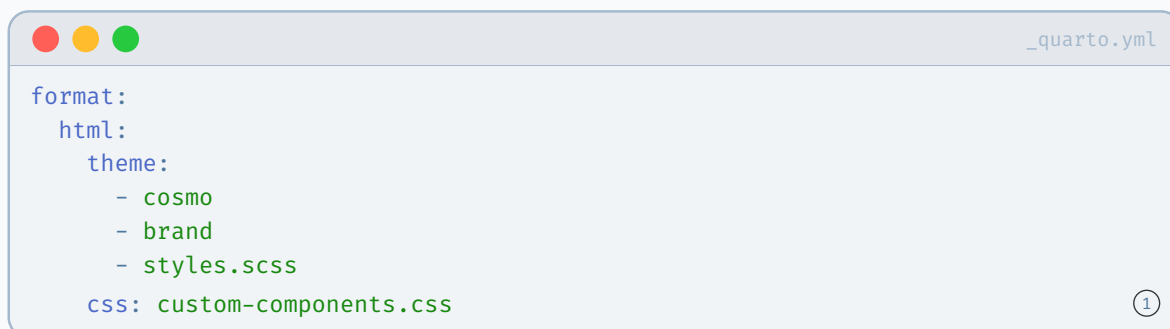
See [Logo Configuration](#) for more.

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:



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

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?
- 4.2. Creating a Brand Extension
 - 4.2.1. Step 1: Scaffold Extension
 - 4.2.2. Step 2: Configure Extension
 - 4.2.3. Step 3: Design Brand System
 - 4.2.4. Step 4: Add Assets
 - 4.2.5. Step 5: Create Example

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

```
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

```
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

```
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"
```

```
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;
    }

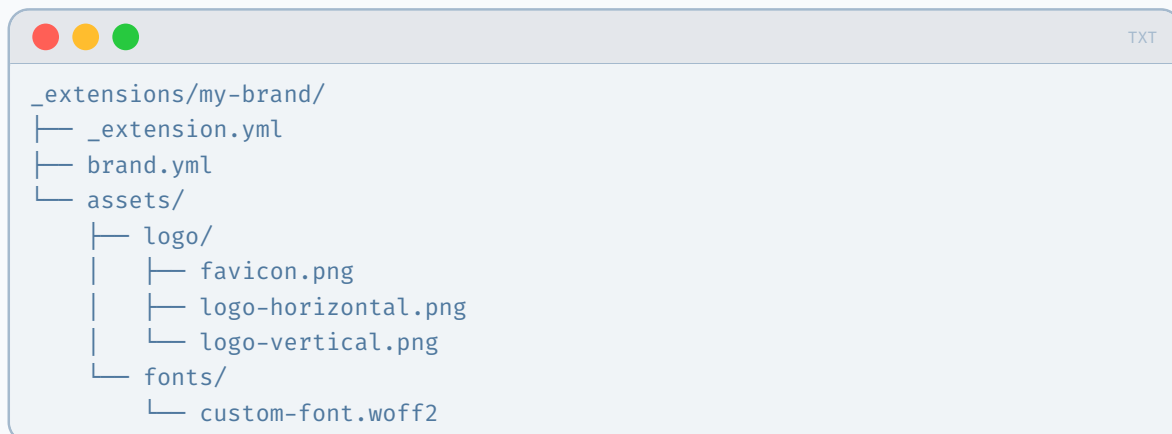
    .btn-primary {
      border-radius: 2rem;
      transition: all 0.3s ease;
    }

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

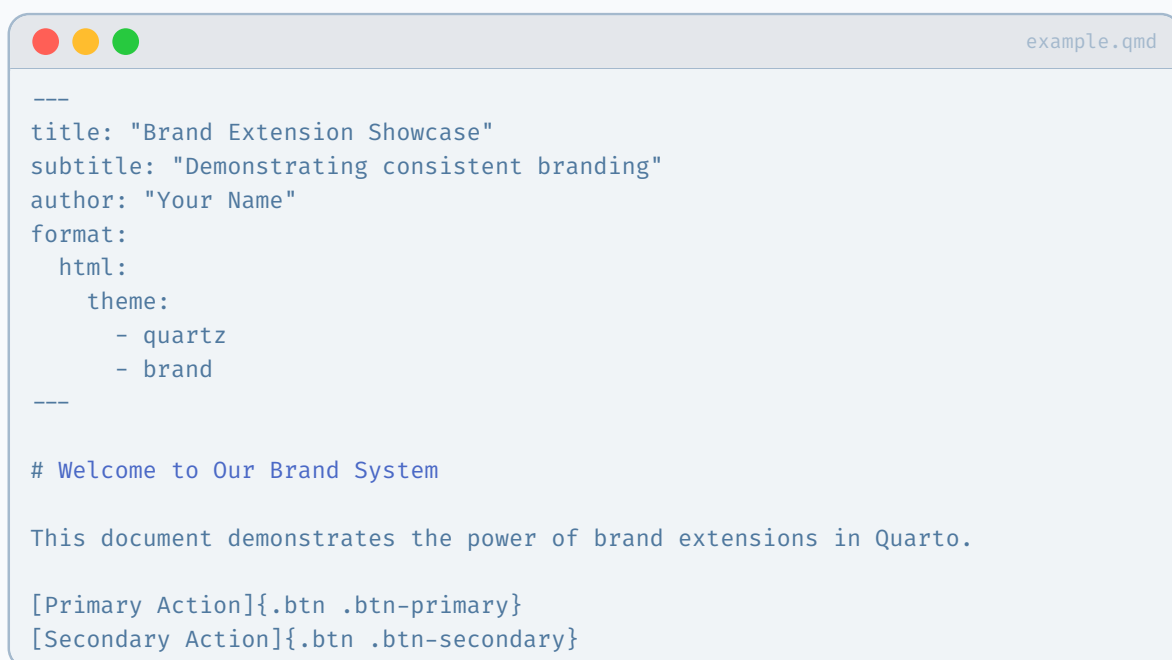
    /* Content styling */
    .title-block-header {
      padding: 3rem 2rem;
      margin-bottom: 2rem;
      border-radius: 0.5rem;
    }
```

4.2.4. Step 4: Add Assets

Organise brand assets:



4.2.5. Step 5: Create Example



! Important

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

5. Hands-On Exercise: Building Your Brand Extension

- 5.1. Exercise Overview
- 5.2. Part 1: Brand Extension Setup
- 5.3. Part 2: Testing
- 5.4. Success Criteria

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](#)

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

5.2. Part 1: Brand Extension Setup

1. Generate extension scaffold:

```
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:

```
quarto create project
```

2. Test across formats:

```
# 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
- 6.2. Basic Variable Syntax
- 6.3. Conditional Content
- 6.4. Loops for Multiple Items
- 6.5. Separator Usage

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

```

---
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:

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

i Note

Use

```
$if(variable)$ ... $else$ ... $endif$
```

for alternative content.

```
Title: $title$

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

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

```
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:

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

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

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


i 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

```

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

Template Input

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

Template Input

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

Template Output

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

Template Output

```
Tags: ResearchData ScienceMachine
Learning
```

7. Hands-On Exercise: Pandoc Templating

- 7.1. Exercise Overview
- 7.2. Part 1: Template Modification
- 7.3. Part 2: Customise Further
- 7.4. Success Criteria

7.1. Exercise Overview

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

Example Code: [Exercises Partial](#)s



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 Partial](#)s
 - [HTML Partial](#)s
 - [Reveal.js Partial](#)s
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
- 8.2. Typst Template Structure
- 8.3. Simple Customisation Example
- 8.4. Creating a Typst Format
 - 8.4.1. Step 1: Scaffold Extension
 - 8.4.2. Step 2: Integration with `brand.yml`

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$sep$, $endfor$],
  $endif$
  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,
)
```

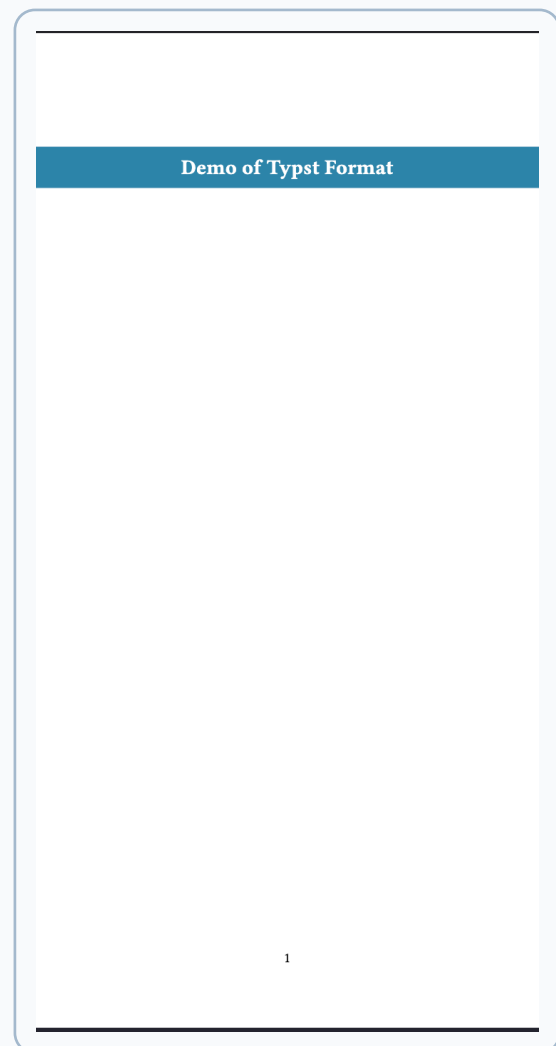


Figure 8.1: Screenshot of a PDF document with blue header titled Demo of Typst Format on white background with page number 1 at bottom

8.4. Creating a Typst Format

8.4.1. Step 1: Scaffold Extension

```
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

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
- 9.2. Part 1: Format Extension Setup
- 9.3. Part 2: Customise and Test Your Format
- 9.4. Part 3: Optional HTML Variant
- 9.5. Success Criteria

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](#)

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

9.2. Part 1: Format Extension Setup

1. Generate format extension scaffold:

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

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

```
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`:

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

3. Render and verify:

```
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:

```
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.