

Formatting & Branding

Session 5

Mickaël CANOUIL, *Ph.D.* 

pro@mickael.canouil.dev

Independent Contractor

Saturday, the 7th of February, 2026

- Session 5: Formatting & Branding
- Session Objectives
- Learning Objectives

Session Overview

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.

- **Pandoc Templating & Extensions**

- Understanding Pandoc's templating syntax fundamentals.
- Variables, conditionals, and loops for dynamic content.
- Developing brand extensions for organisational reuse.

- **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.

- **Typst Customisation**

- Typst templates and partials for PDF branding.
- Template structure and brand.yml integration.
- Creating custom Typst formats with consistent visual identity.

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.

Learning Objectives

Learning Objectives

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

Learning Objectives

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

- Implement professional themes with Bootstrap 5 customisation.

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.

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`.

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.

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.

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.

- Bootstrap in Quarto
- Theme Selection
- Basic Customisation
- Progressive Enhancement

Theming (Bootstrap & Customisation)

Bootstrap in Quarto

Bootstrap in Quarto

Quarto HTML format uses Bootstrap 5, providing:

Bootstrap in Quarto

Quarto HTML format uses Bootstrap 5, providing:

- **Several built-in themes** from Bootswatch.

Bootstrap in Quarto

Quarto HTML format uses Bootstrap 5, providing:

- **Several built-in themes** from Bootswatch.
- **Responsive grid system.**

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).

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.

Theme Selection

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.

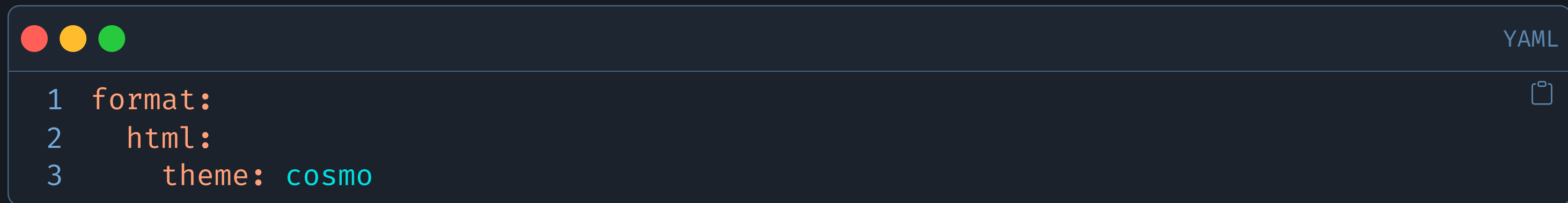
Theme Selection

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.

Theme Selection

- Single theme.

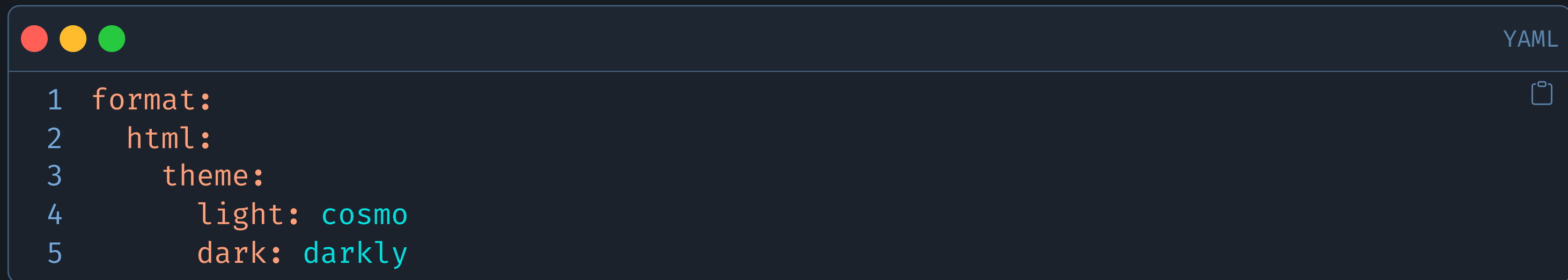


```
1 format:
2   html:
3     theme: cosmo
```

The image shows a terminal window with a dark background. The title bar at the top has three colored circles (red, yellow, green) on the left and the text 'YAML' on the right. The terminal content shows a YAML configuration with three lines: '1 format:', '2 html:', and '3 theme: cosmo'. The word 'cosmo' is highlighted in cyan. A small clipboard icon is visible on the right side of the terminal window.

Theme Selection

- Light/dark theme pair.



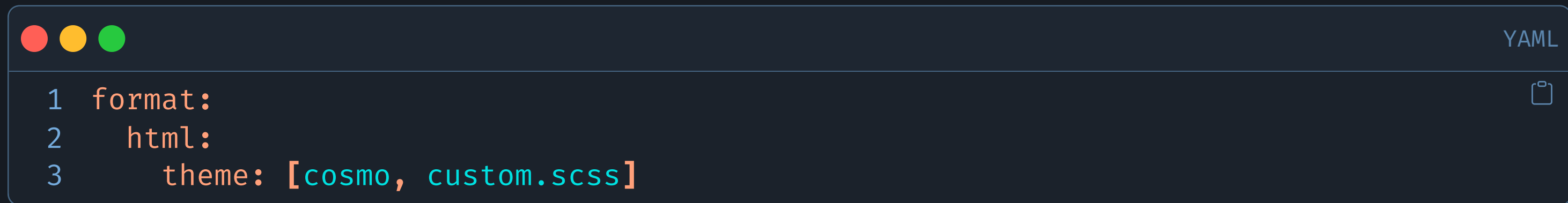
A code editor window with a dark theme. The title bar shows three colored circles (red, yellow, green) on the left and the text 'YAML' on the right. The editor contains the following YAML code:

```
1 format:
2   html:
3     theme:
4       light: cosmo
5       dark: darkly
```

A small clipboard icon is visible in the top right corner of the editor area.

Theme Selection

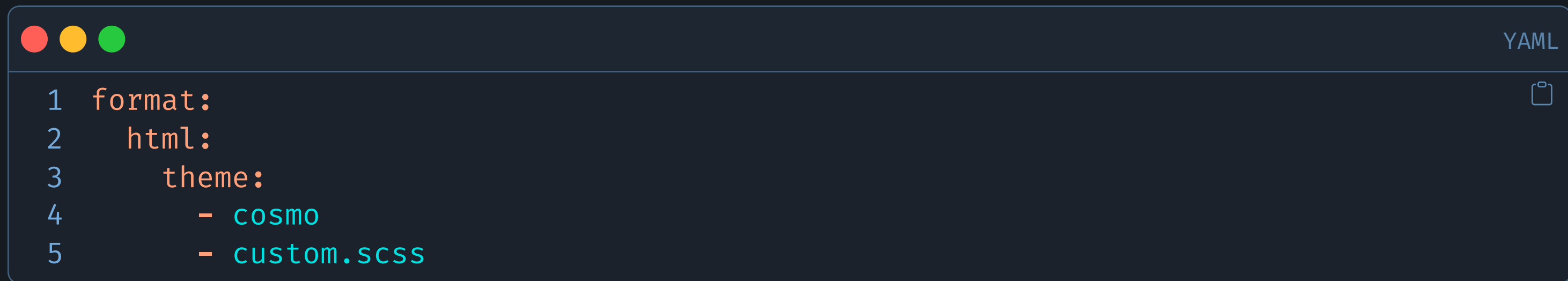
- Theme + customisation.



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

Theme Selection

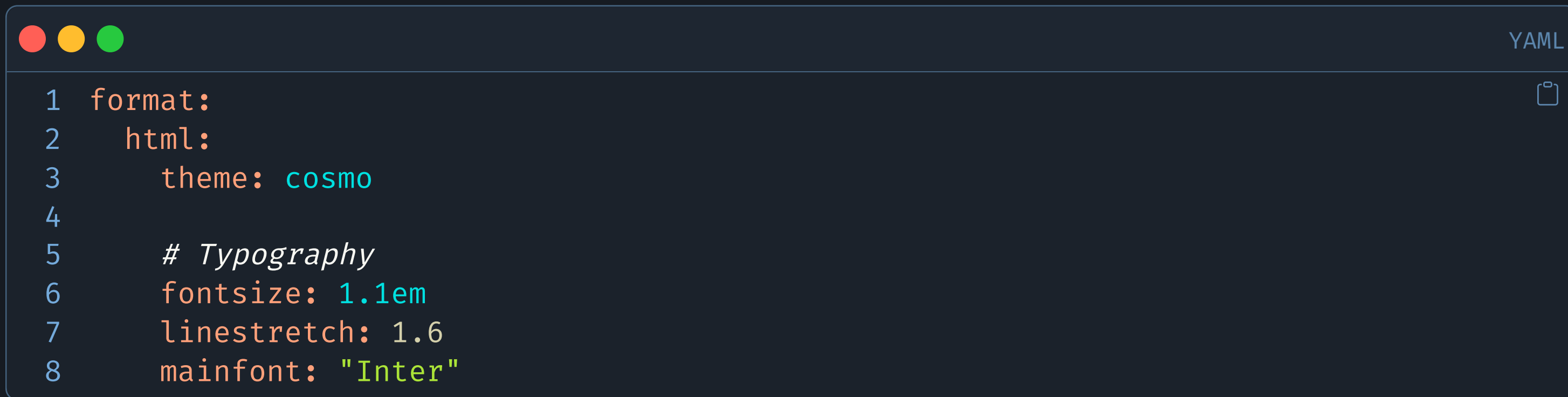
Can also be written as:



```
1 format:
2   html:
3     theme:
4       - cosmo
5       - custom.scss
```

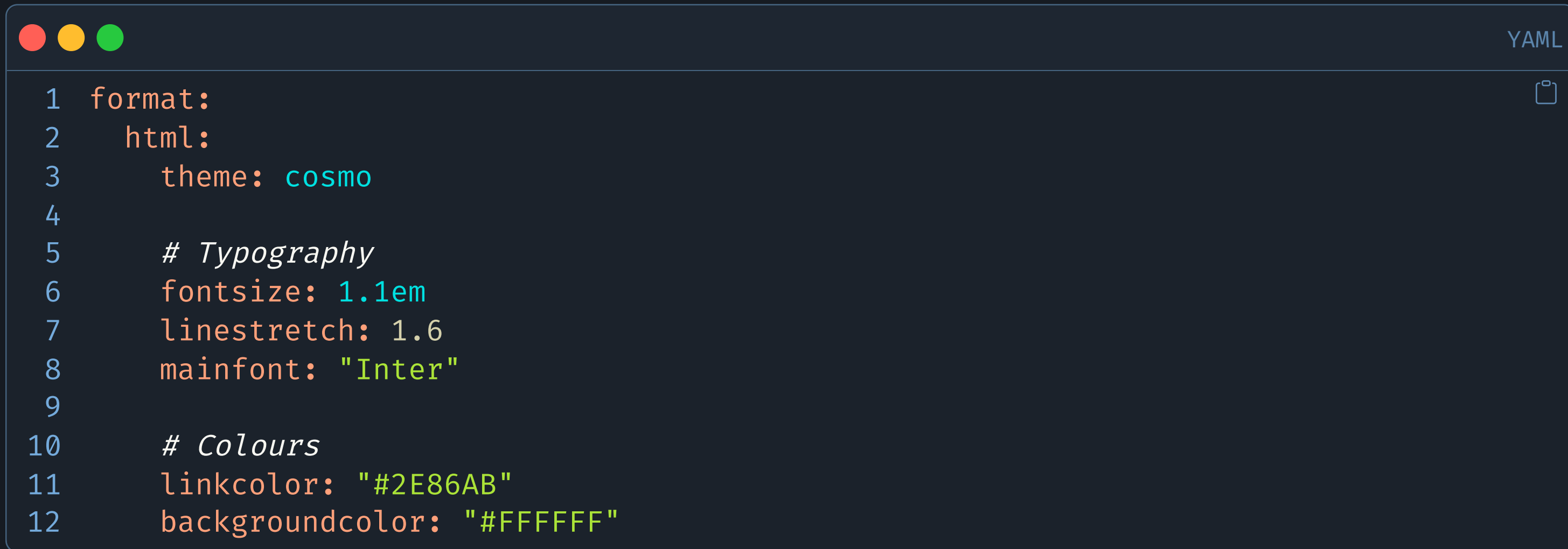
The code editor window has a title bar with three colored circles (red, yellow, green) on the left and the text 'YAML' on the right. A clipboard icon is visible in the top right corner of the editor area.

Basic Customisation



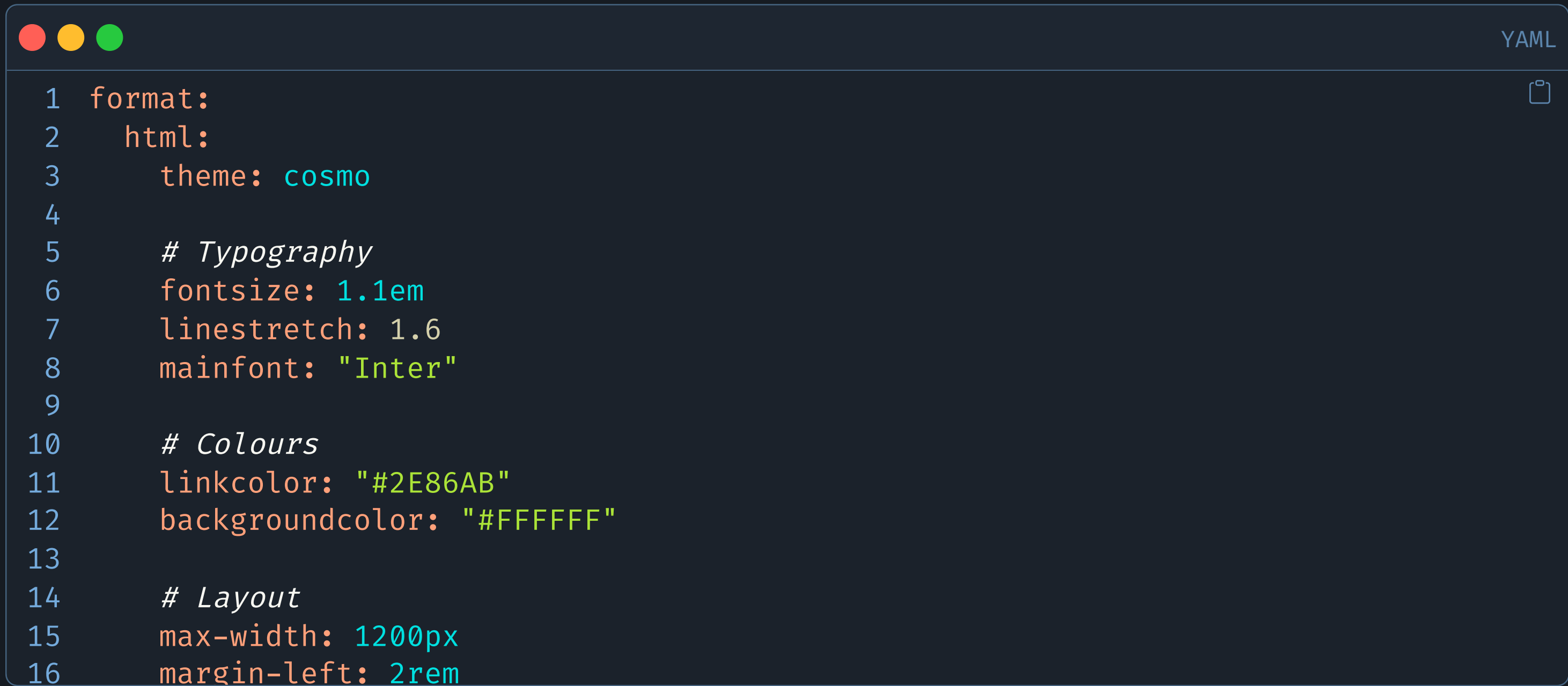
```
1 format:
2   html:
3     theme: cosmo
4
5     # Typography
6     fontsize: 1.1em
7     linestretch: 1.6
8     mainfont: "Inter"
```

Basic Customisation



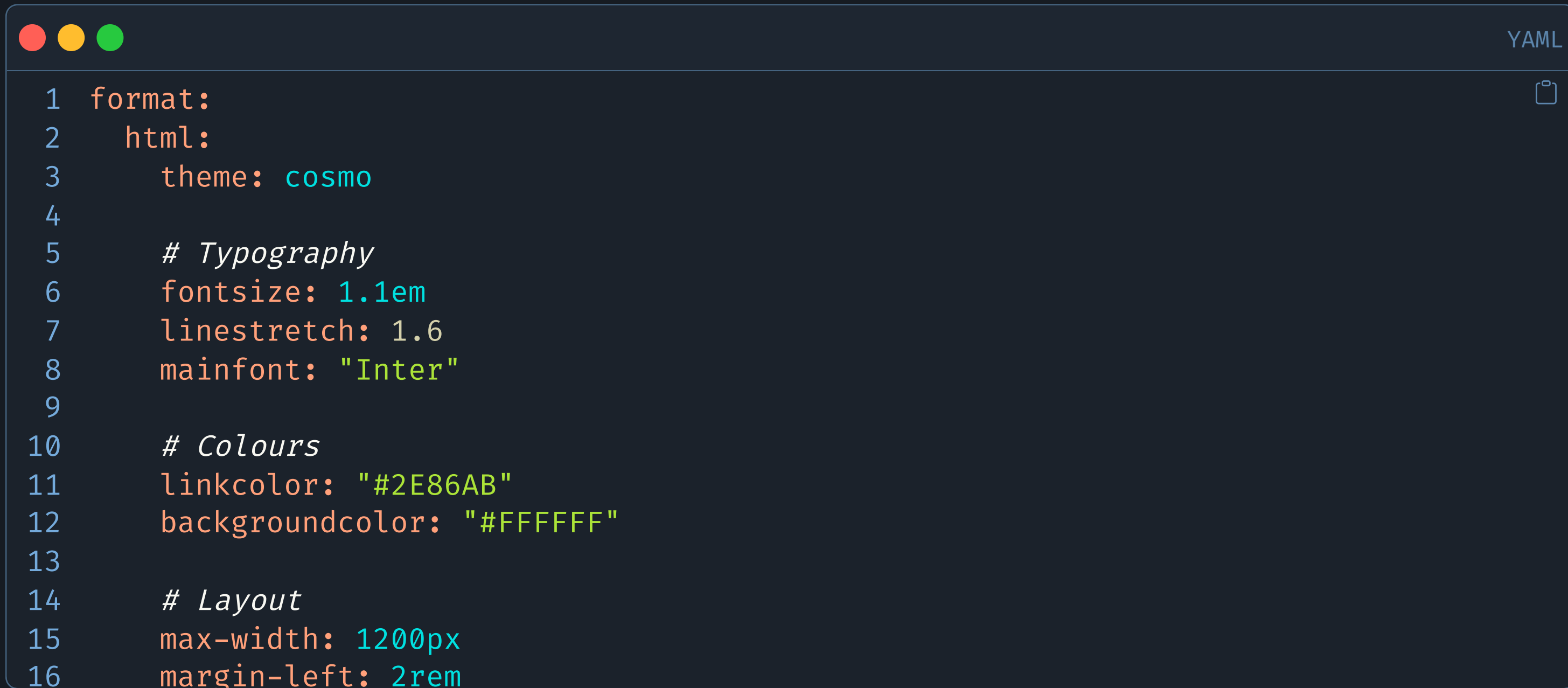
```
1 format:
2   html:
3     theme: cosmo
4
5     # Typography
6     fontsize: 1.1em
7     linestretch: 1.6
8     mainfont: "Inter"
9
10    # Colours
11    linkcolor: "#2E86AB"
12    backgroundcolor: "#FFFFFFF"
```

Basic Customisation



```
1 format:
2   html:
3     theme: cosmo
4
5     # Typography
6     fontsize: 1.1em
7     linestretch: 1.6
8     mainfont: "Inter"
9
10    # Colours
11    linkcolor: "#2E86AB"
12    backgroundcolor: "#FFFFFF"
13
14    # Layout
15    max-width: 1200px
16    margin-left: 2rem
```


Basic Customisation

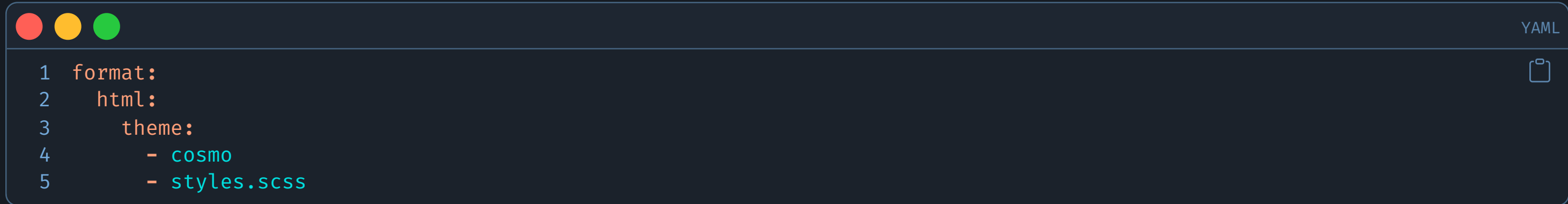


```
1 format:
2   html:
3     theme: cosmo
4
5     # Typography
6     fontsize: 1.1em
7     linestretch: 1.6
8     mainfont: "Inter"
9
10    # Colours
11    linkcolor: "#2E86AB"
12    backgroundcolor: "#FFFFFF"
13
14    # Layout
15    max-width: 1200px
16    margin-left: 2rem
```

See [HTML Theming - Basic Options](#) for more.

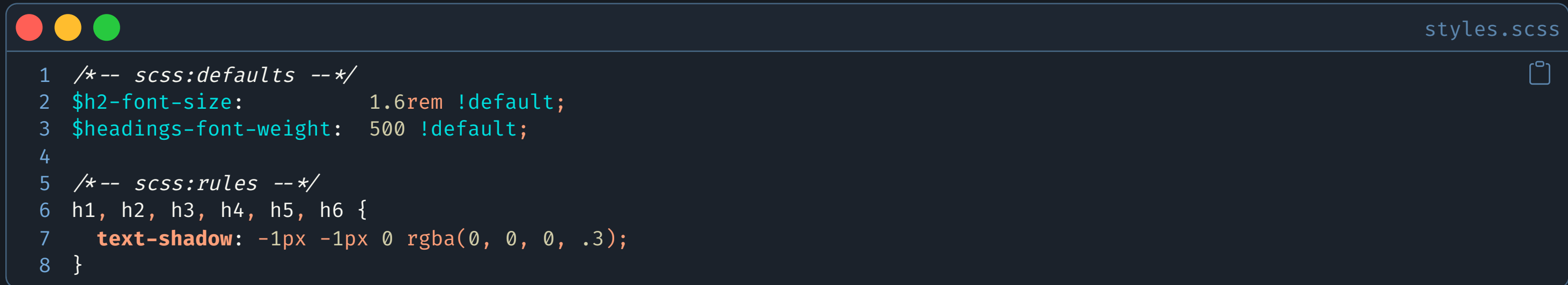
Progressive Enhancement

Move to SCSS for more control:



```
1 format:
2   html:
3     theme:
4       - cosmo
5       - styles.scss
```

Custom SCSS file follows a specific structure using comments:



```
1 /*-- scss:defaults --*/
2 $h2-font-size: 1.6rem !default;
3 $headings-font-weight: 500 !default;
4
5 /*-- scss:rules --*/
6 h1, h2, h3, h4, h5, h6 {
7   text-shadow: -1px -1px 0 rgba(0, 0, 0, .3);
8 }
```

See [More About Quarto Themes](#) for more.

- Understanding Brand.yml
- Basic brand.yml Structure
- Colour Palette System
- Typography Configuration
- Logo Management
- Brand Integration

brand.yml Foundation

Understanding Brand.yml

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.

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:

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.

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.

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.

Understanding Brand.yml

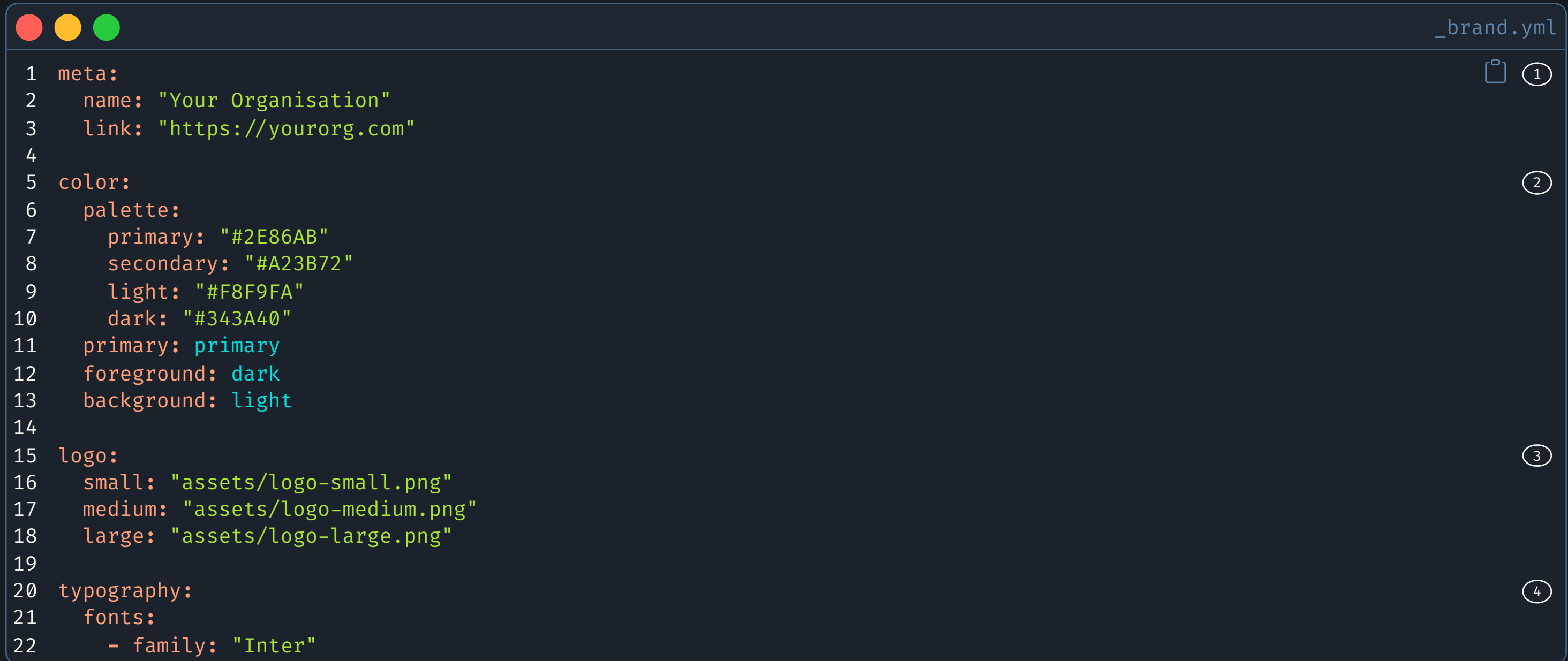
`brand.yml` is a single YAML file that codifies your organisation's brand guidelines into a format that can be used by Quarto (*but not limited to*) to create branded outputs across multiple formats.

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.

Basic `brand.yml` Structure

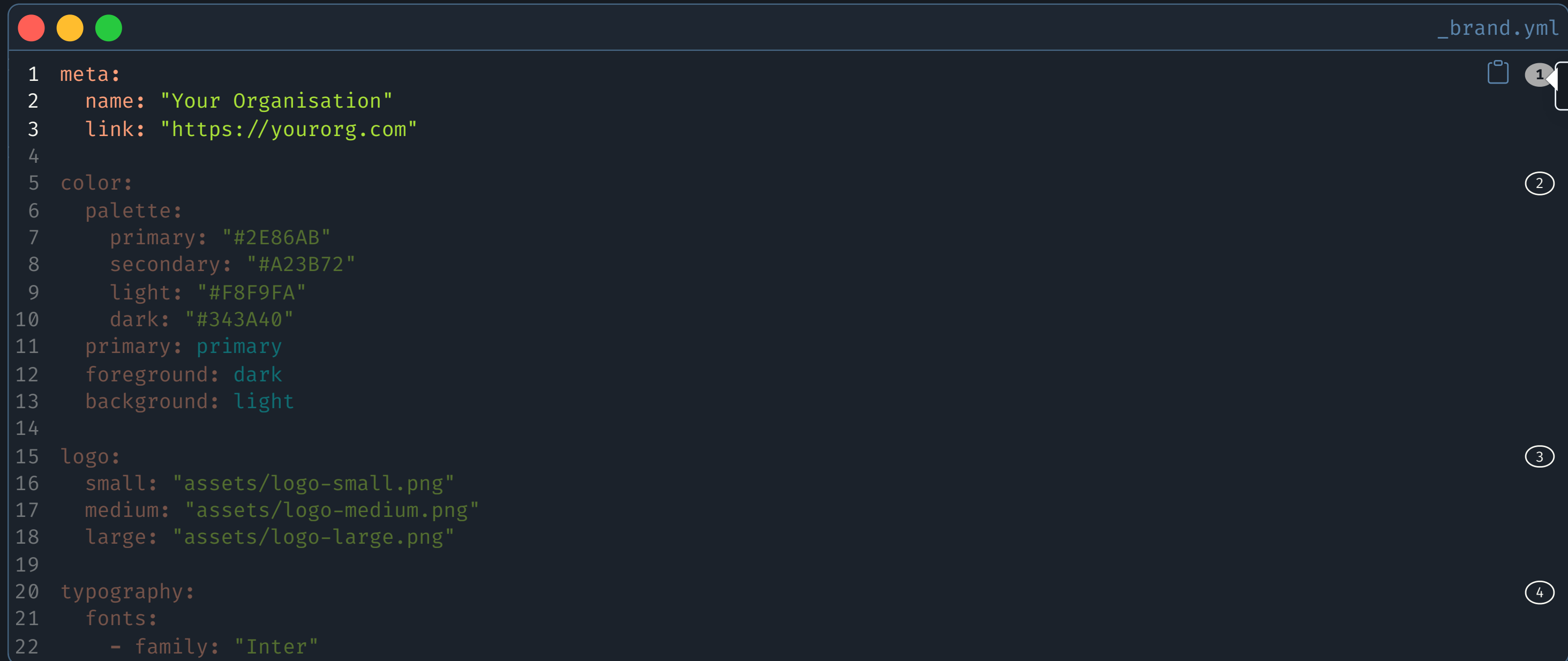
Create a `_brand.yml` file in your project root:



```
1 meta:
2   name: "Your Organisation"
3   link: "https://yourorg.com"
4
5 color:
6   palette:
7     primary: "#2E86AB"
8     secondary: "#A23B72"
9     light: "#F8F9FA"
10    dark: "#343A40"
11   primary: primary
12   foreground: dark
13   background: light
14
15 logo:
16   small: "assets/logo-small.png"
17   medium: "assets/logo-medium.png"
18   large: "assets/logo-large.png"
19
20 typography:
21   fonts:
22     - family: "Inter"
```


Basic `brand.yml` Structure

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

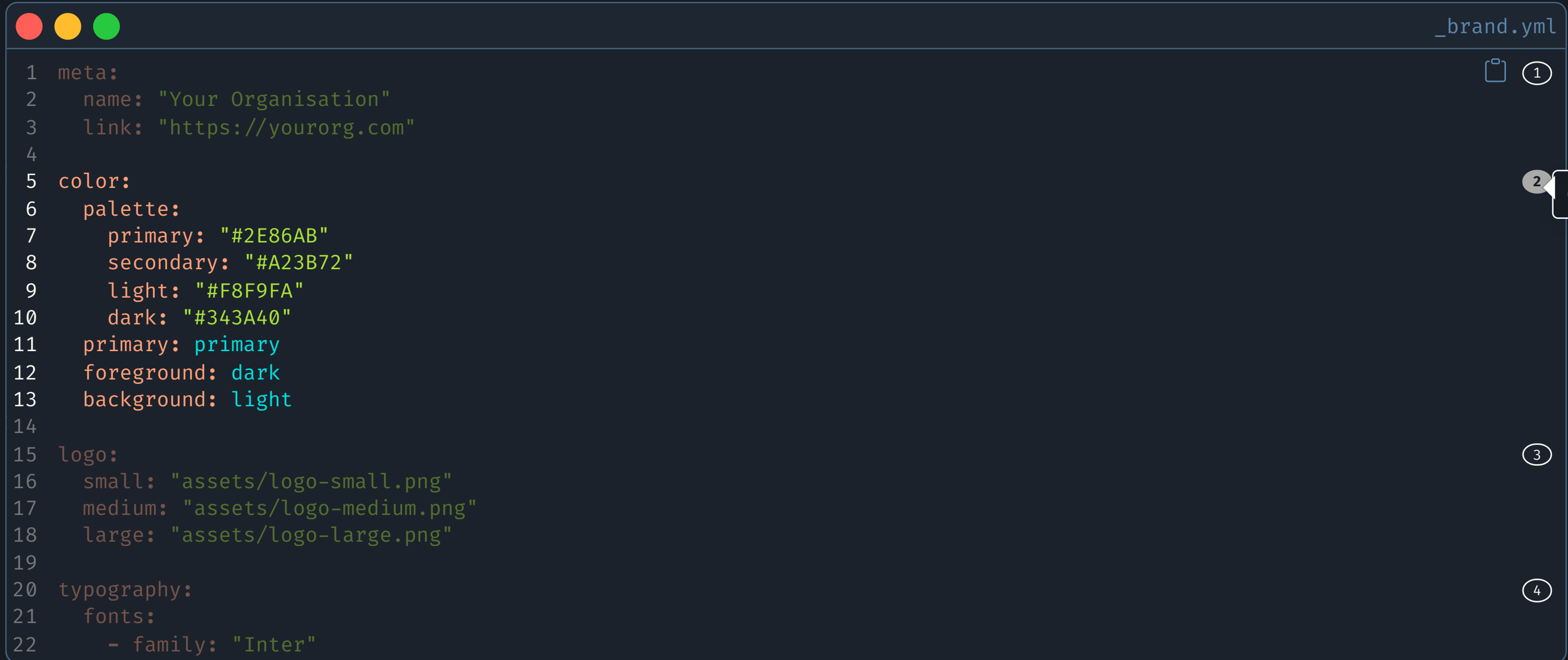


```
1 meta:
2   name: "Your Organisation"
3   link: "https://yourorg.com"
4
5 color:
6   palette:
7     primary: "#2E86AB"
8     secondary: "#A23B72"
9     light: "#F8F9FA"
10    dark: "#343A40"
11    primary: primary
12    foreground: dark
13    background: light
14
15 logo:
16   small: "assets/logo-small.png"
17   medium: "assets/logo-medium.png"
18   large: "assets/logo-large.png"
19
20 typography:
21   fonts:
22     - family: "Inter"
```

Basic brand identity.

Basic `brand.yml` Structure

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

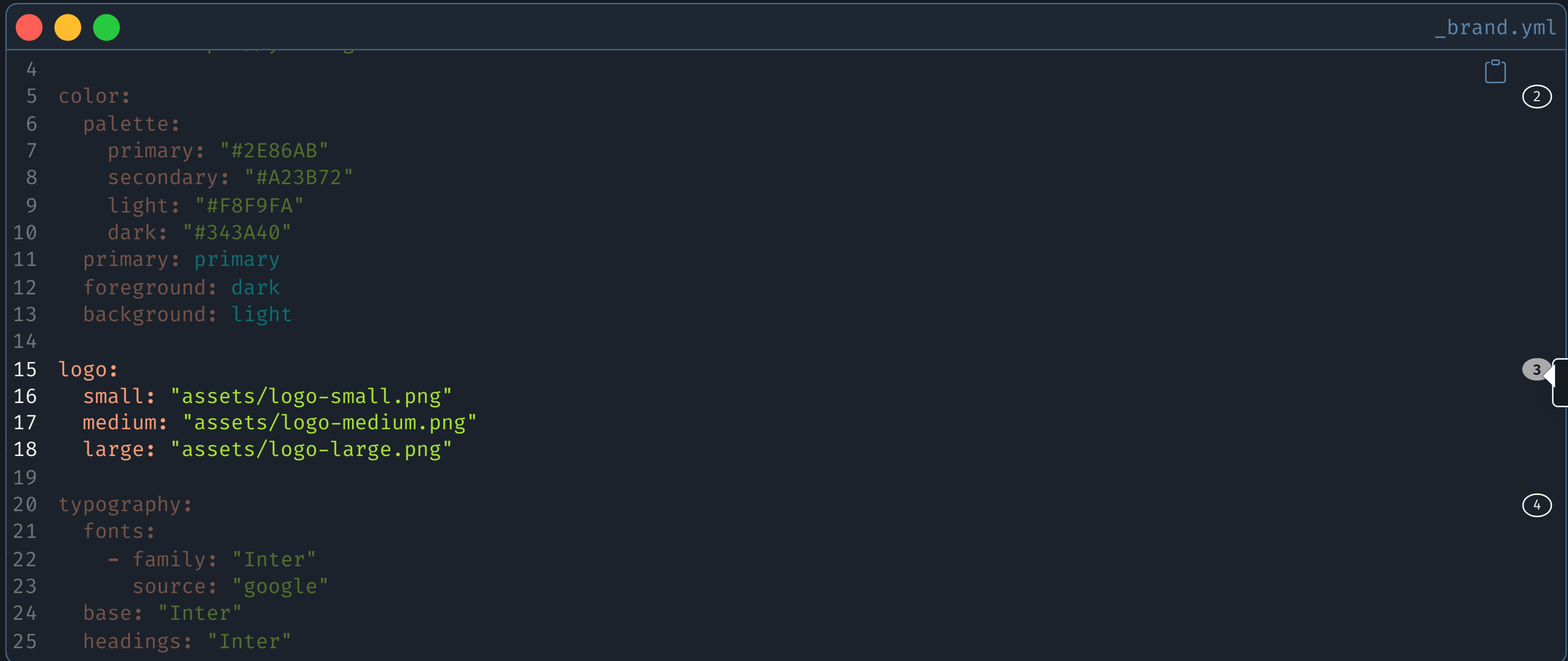


```
1 meta:
2   name: "Your Organisation"
3   link: "https://yourorg.com"
4
5 color:
6   palette:
7     primary: "#2E86AB"
8     secondary: "#A23B72"
9     light: "#F8F9FA"
10    dark: "#343A40"
11   primary: primary
12   foreground: dark
13   background: light
14
15 logo:
16   small: "assets/logo-small.png"
17   medium: "assets/logo-medium.png"
18   large: "assets/logo-large.png"
19
20 typography:
21   fonts:
22     - family: "Inter"
```

Colour palette system.

Basic `brand.yml` Structure

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

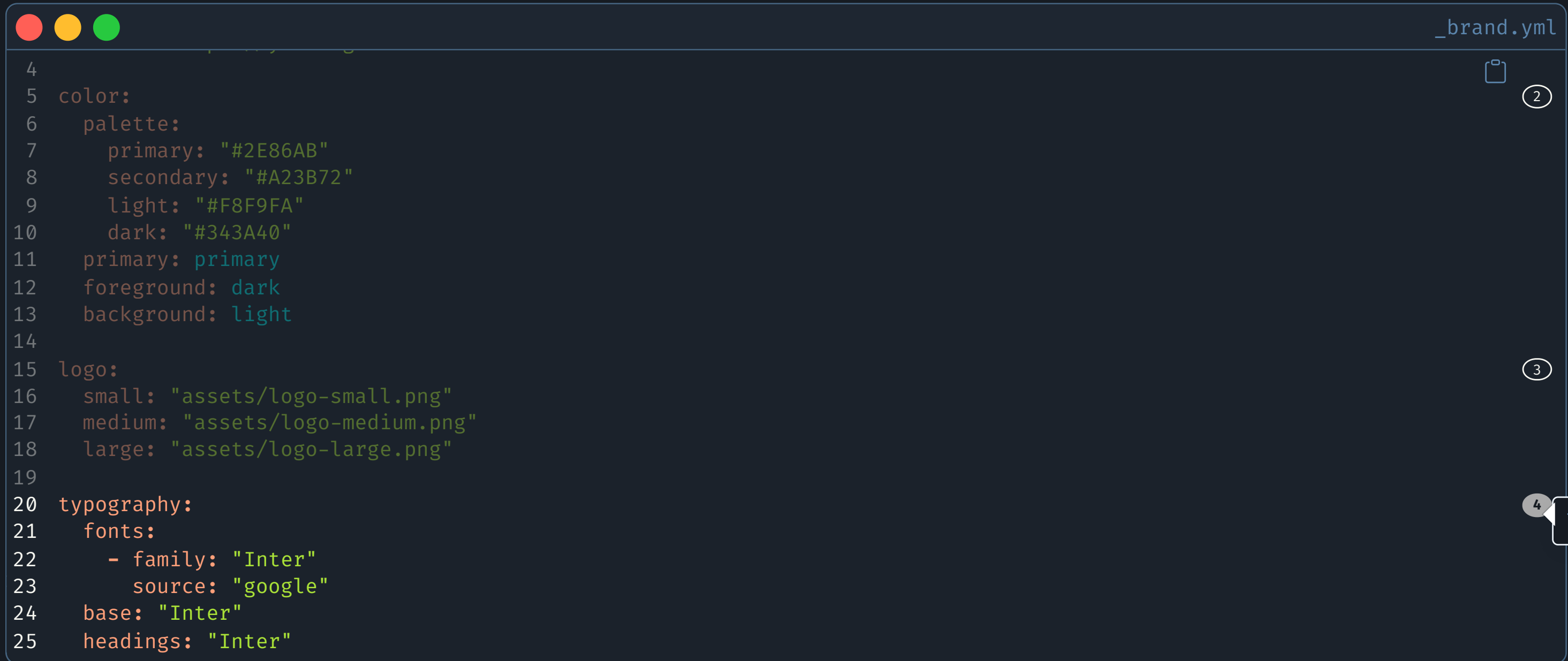


```
4  
5 color:  
6   palette:  
7     primary: "#2E86AB"  
8     secondary: "#A23B72"  
9     light: "#F8F9FA"  
10    dark: "#343A40"  
11    primary: primary  
12    foreground: dark  
13    background: light  
14  
15 logo:  
16   small: "assets/logo-small.png"  
17   medium: "assets/logo-medium.png"  
18   large: "assets/logo-large.png"  
19  
20 typography:  
21   fonts:  
22     - family: "Inter"  
23       source: "google"  
24   base: "Inter"  
25   headings: "Inter"
```

The screenshot shows a code editor window titled `_brand.yml`. The code is a YAML file defining brand settings. It includes a `color` section with a `palette` of colors and a `logo` section with three image paths. The `typography` section defines fonts, including a list of font families and a base font. The code is syntax-highlighted. On the right side of the editor, there are two numbered callouts: a clipboard icon with a circled '2' and a callout box with a circled '3' containing the text 'Logo management.' and a circled '4'.

Basic `brand.yml` Structure

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

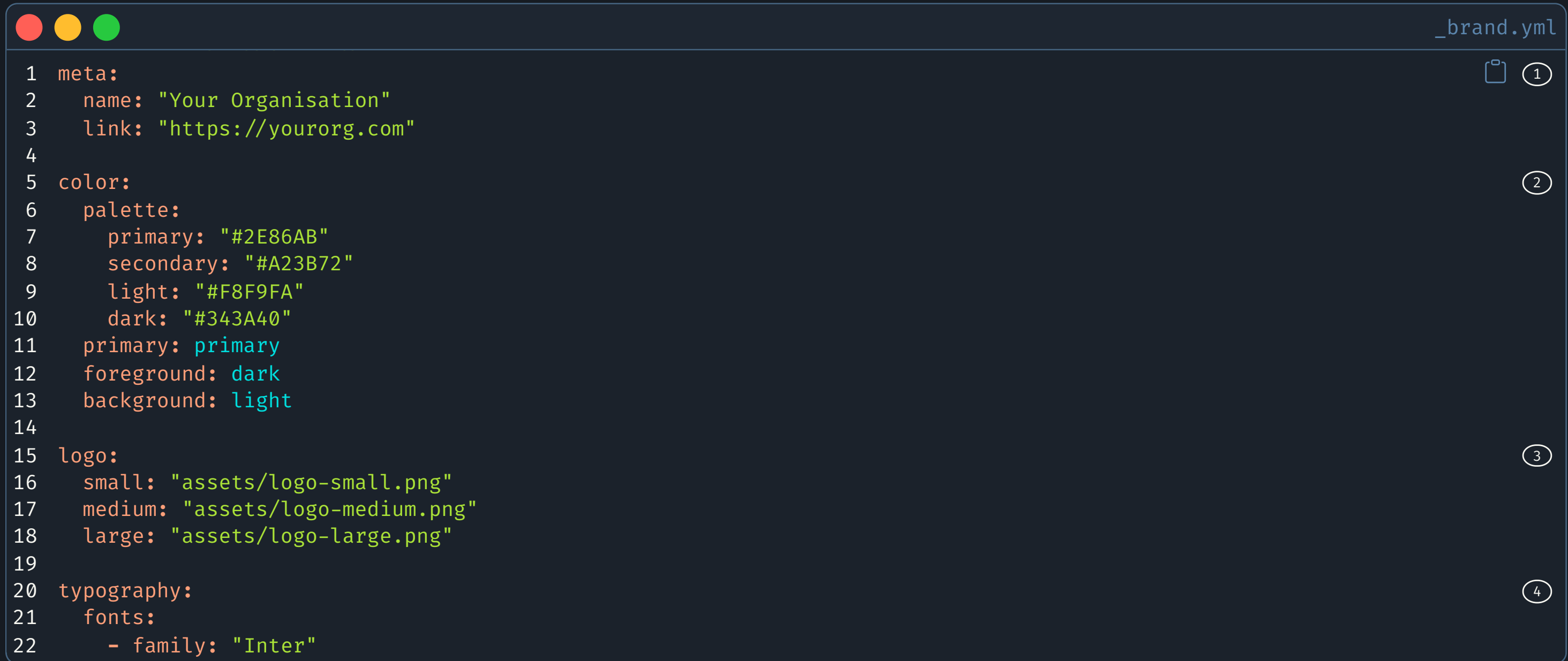


```
4
5 color:
6   palette:
7     primary: "#2E86AB"
8     secondary: "#A23B72"
9     light: "#F8F9FA"
10    dark: "#343A40"
11    primary: primary
12    foreground: dark
13    background: light
14
15 logo:
16   small: "assets/logo-small.png"
17   medium: "assets/logo-medium.png"
18   large: "assets/logo-large.png"
19
20 typography:
21   fonts:
22     - family: "Inter"
23       source: "google"
24   base: "Inter"
25   headings: "Inter"
```

4 Typography system.

Basic `brand.yml` Structure

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

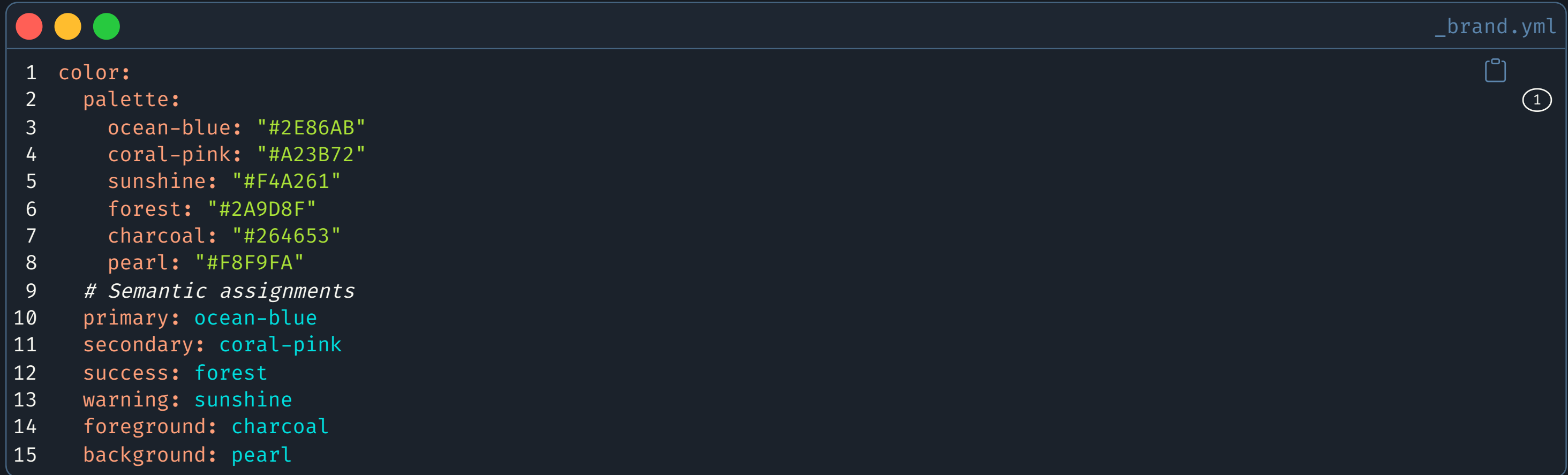


```
1 meta:
2   name: "Your Organisation"
3   link: "https://yourorg.com"
4
5 color:
6   palette:
7     primary: "#2E86AB"
8     secondary: "#A23B72"
9     light: "#F8F9FA"
10    dark: "#343A40"
11   primary: primary
12   foreground: dark
13   background: light
14
15 logo:
16   small: "assets/logo-small.png"
17   medium: "assets/logo-medium.png"
18   large: "assets/logo-large.png"
19
20 typography:
21   fonts:
22     - family: "Inter"
```


Colour Palette System

Colour Palette System

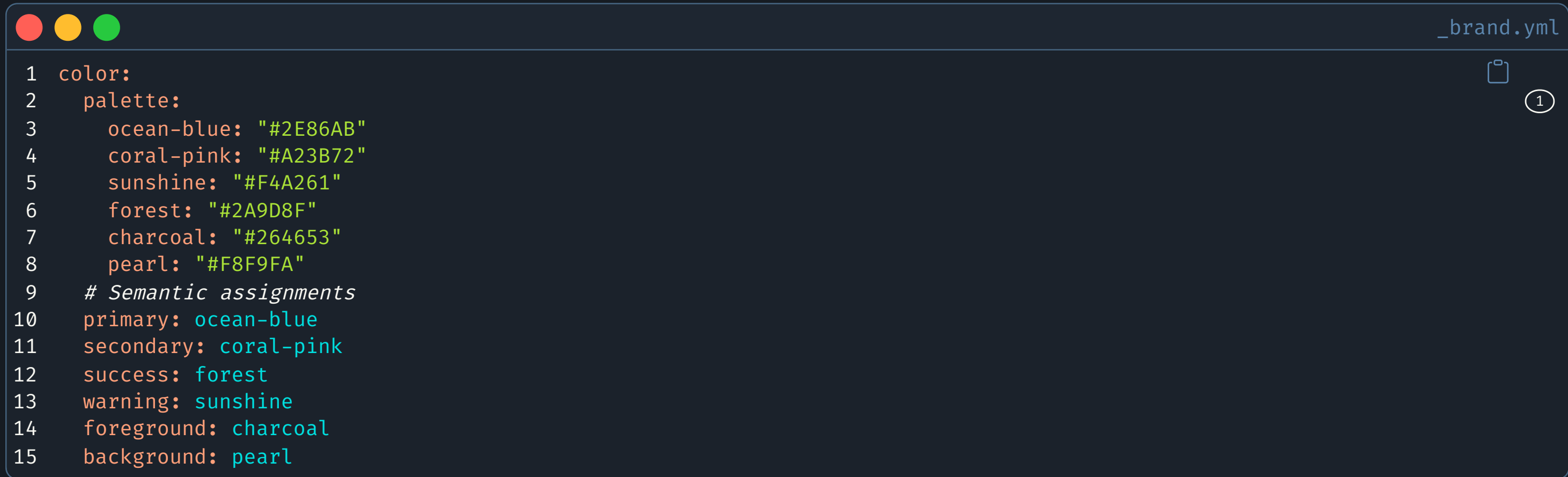
The palette defines named colours that can be referenced throughout your brand:



```
1 color:
2   palette:
3     ocean-blue: "#2E86AB"
4     coral-pink: "#A23B72"
5     sunshine: "#F4A261"
6     forest: "#2A9D8F"
7     charcoal: "#264653"
8     pearl: "#F8F9FA"
9   # Semantic assignments
10  primary: ocean-blue
11  secondary: coral-pink
12  success: forest
13  warning: sunshine
14  foreground: charcoal
15  background: pearl
```

Colour Palette System

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

A code editor window with a dark theme. The title bar shows three colored circles (red, yellow, green) on the left and the filename "_brand.yml" on the right. The editor contains 15 lines of YAML code. Lines 1-8 define a color palette with six entries: ocean-blue, coral-pink, sunshine, forest, charcoal, and pearl, each with a hex color value. Line 9 is a comment. Lines 10-15 define semantic assignments for primary, secondary, success, warning, foreground, and background colors, each referencing one of the palette names. A clipboard icon and a circled number 1 are visible in the top right corner of the editor area.

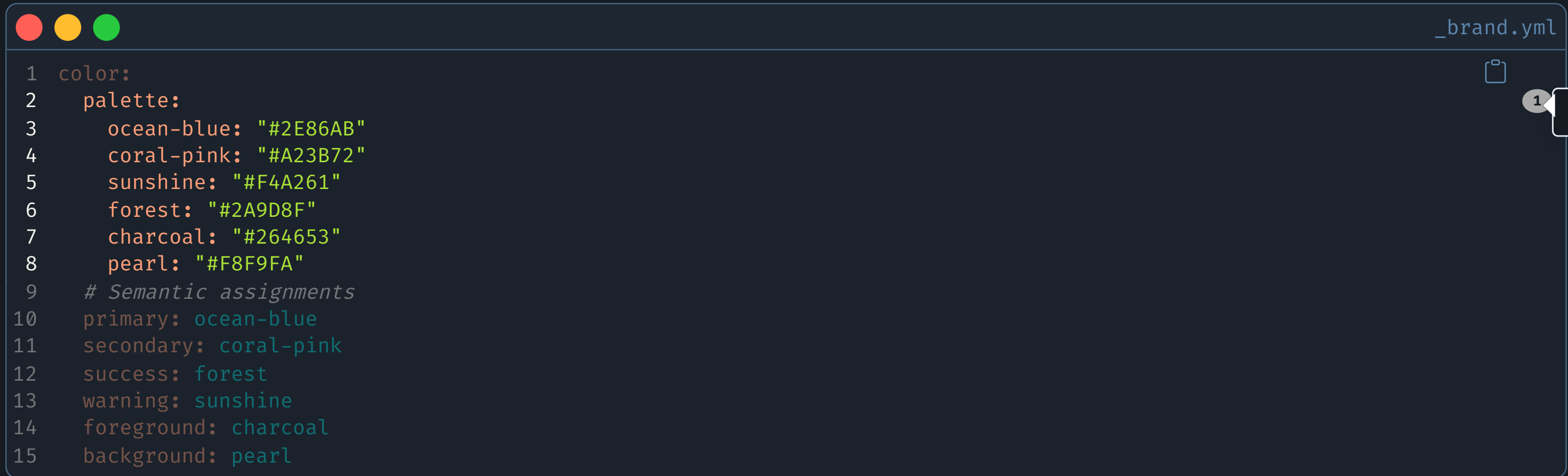
```
1 color:
2   palette:
3     ocean-blue: "#2E86AB"
4     coral-pink: "#A23B72"
5     sunshine: "#F4A261"
6     forest: "#2A9D8F"
7     charcoal: "#264653"
8     pearl: "#F8F9FA"
9   # Semantic assignments
10  primary: ocean-blue
11  secondary: coral-pink
12  success: forest
13  warning: sunshine
14  foreground: charcoal
15  background: pearl
```

Cross-format usage:

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

Colour Palette System

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



```
1 color:
2   palette:
3     ocean-blue: "#2E86AB"
4     coral-pink: "#A23B72"
5     sunshine: "#F4A261"
6     forest: "#2A9D8F"
7     charcoal: "#264653"
8     pearl: "#F8F9FA"
9   # Semantic assignments
10  primary: ocean-blue
11  secondary: coral-pink
12  success: forest
13  warning: sunshine
14  foreground: charcoal
15  background: pearl
```

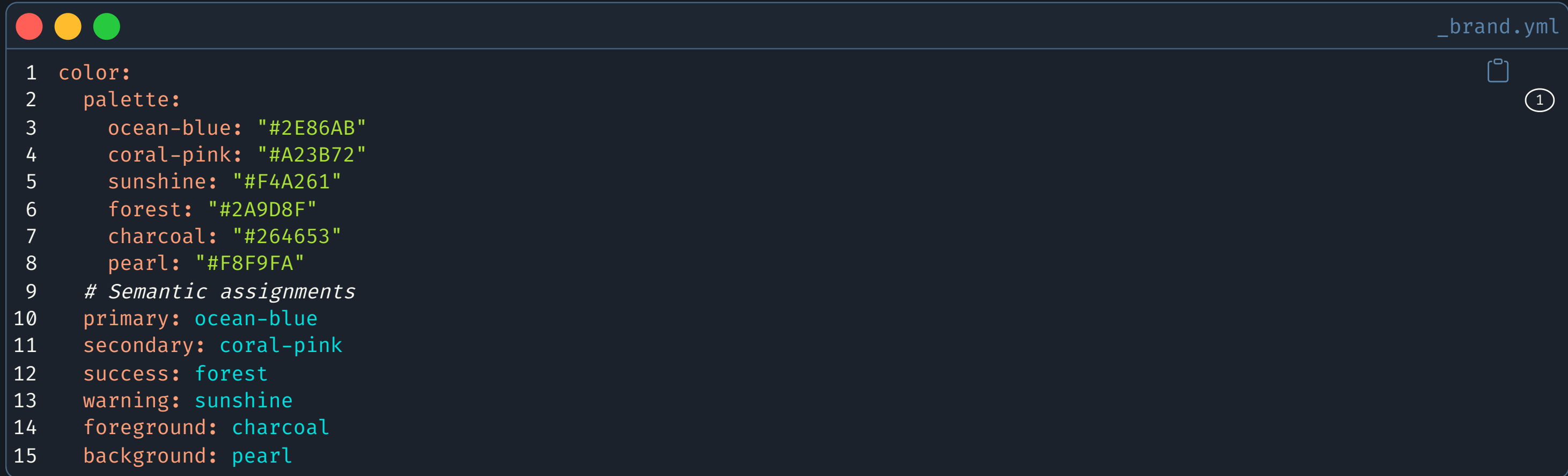
The code editor window displays a file named `_brand.yml`. The content is a YAML configuration for a color palette. It starts with a `color:` key, followed by a `palette:` key. Under `palette:`, there are six entries, each with a name and a hex color code: `ocean-blue: "#2E86AB"`, `coral-pink: "#A23B72"`, `sunshine: "#F4A261"`, `forest: "#2A9D8F"`, `charcoal: "#264653"`, and `pearl: "#F8F9FA"`. Below these, there is a comment `# Semantic assignments` followed by six more entries: `primary: ocean-blue`, `secondary: coral-pink`, `success: forest`, `warning: sunshine`, `foreground: charcoal`, and `background: pearl`. A tooltip with the number '1' points to the `palette:` key, with the text "Define a palette of named co".

Cross-format usage:

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

Colour Palette System

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

A code editor window with a dark theme. The title bar shows three colored circles (red, yellow, green) on the left and the filename "_brand.yml" on the right. The editor contains 15 lines of YAML code. Lines 1-8 define a color palette with six entries: ocean-blue, coral-pink, sunshine, forest, charcoal, and pearl, each with a hex color code. Line 9 is a comment. Lines 10-15 define semantic assignments for primary, secondary, success, warning, foreground, and background colors, each referencing a color from the palette. A clipboard icon and a circled number 1 are visible in the top right corner of the editor area.

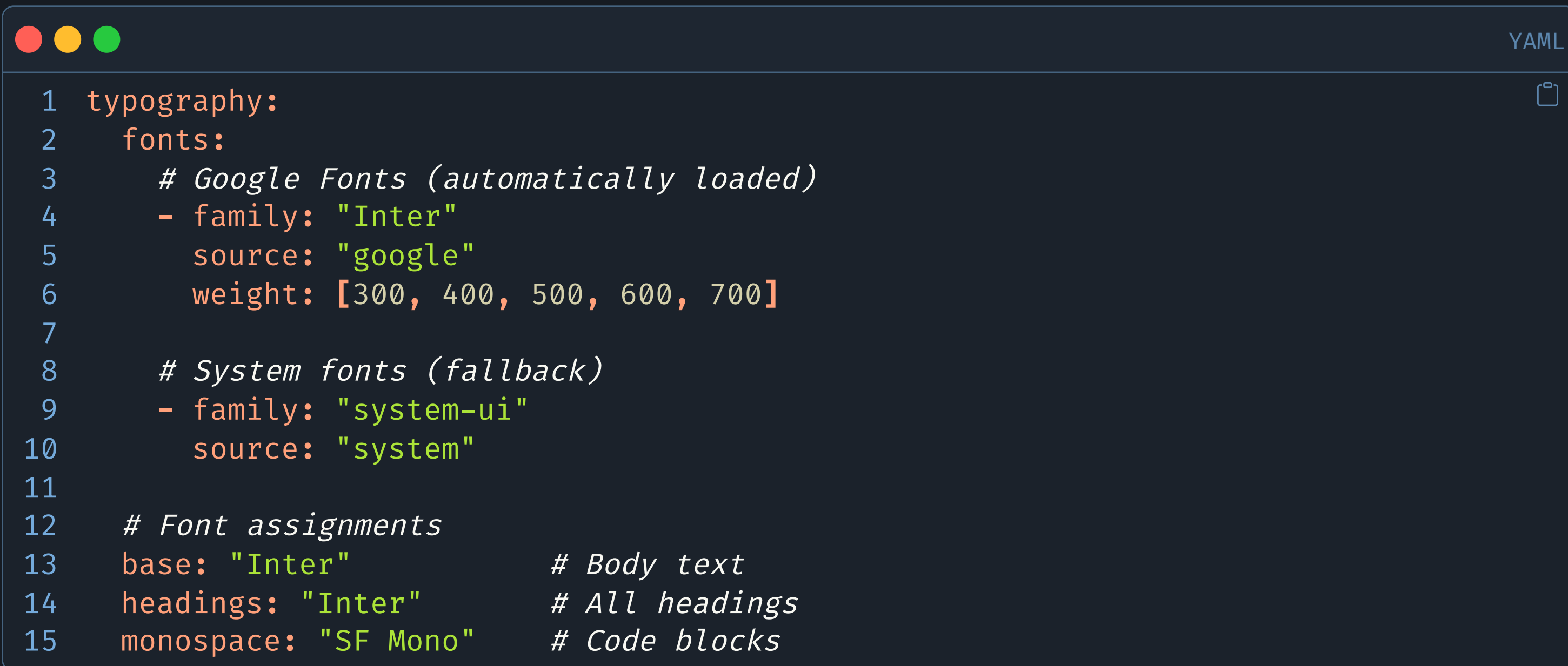
```
1 color:
2   palette:
3     ocean-blue: "#2E86AB"
4     coral-pink: "#A23B72"
5     sunshine: "#F4A261"
6     forest: "#2A9D8F"
7     charcoal: "#264653"
8     pearl: "#F8F9FA"
9   # Semantic assignments
10  primary: ocean-blue
11  secondary: coral-pink
12  success: forest
13  warning: sunshine
14  foreground: charcoal
15  background: pearl
```

Cross-format usage:

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

Typography Configuration

Define your organisation's font system:



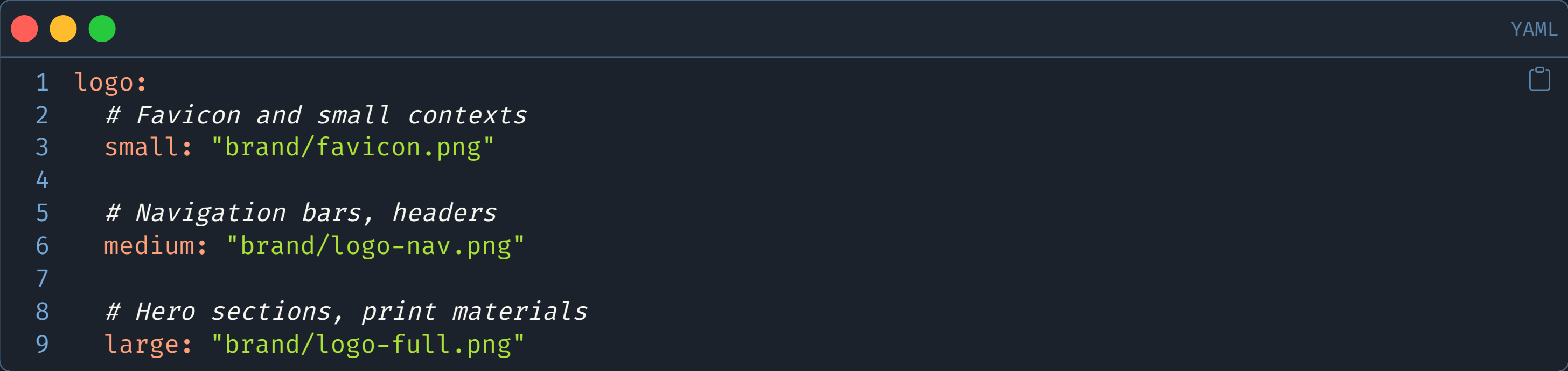
```
1 typography:
2   fonts:
3     # Google Fonts (automatically loaded)
4     - family: "Inter"
5       source: "google"
6       weight: [300, 400, 500, 600, 700]
7
8     # System fonts (fallback)
9     - family: "system-ui"
10      source: "system"
11
12   # Font assignments
13   base: "Inter"           # Body text
14   headings: "Inter"       # All headings
15   monospace: "SF Mono"    # Code blocks
```

See [Typst Brand YAML](#) for troubleshooting fonts in Typst.

Logo Management

Logo Management

Provide logos at different sizes for various contexts:



```
1 logo:
2   # Favicon and small contexts
3   small: "brand/favicon.png"
4
5   # Navigation bars, headers
6   medium: "brand/logo-nav.png"
7
8   # Hero sections, print materials
9   large: "brand/logo-full.png"
```


Logo Management

Provide logos at different sizes for various contexts:

YAML

```
1 logo:
2   # Favicon and small contexts
3   small: "brand/favicon.png"
4
5   # Navigation bars, headers
6   medium: "brand/logo-nav.png"
7
8   # Hero sections, print materials
9   large: "brand/logo-full.png"
```

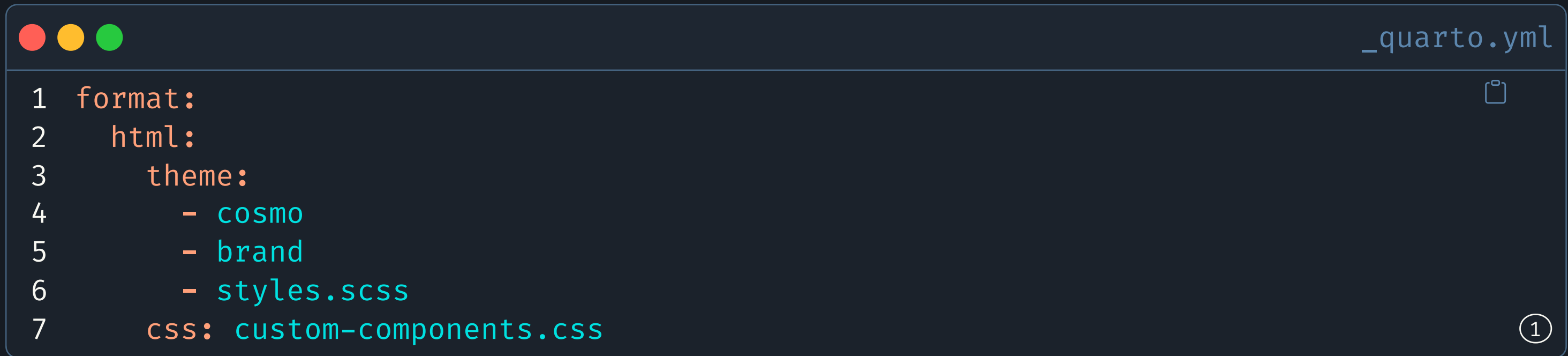
Logo usage across formats.

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 <code>format: typst: logo</code>	medium > small > large
revealjs	Bottom right corner of slides	medium > small > large
website / book project	<code>favicon</code> shown in browser tab	small

See [Logo Configuration](#) for more.

Brand Integration

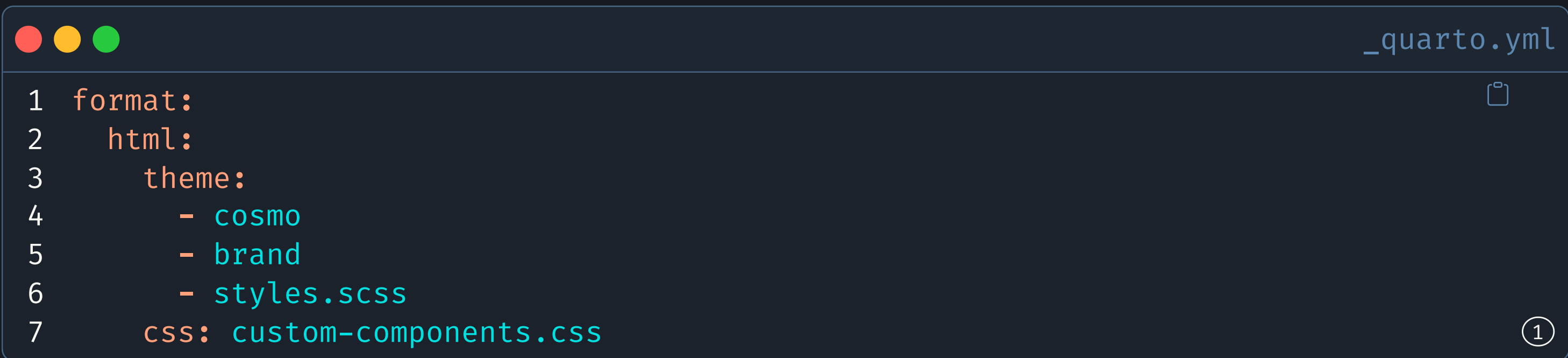
Combine `brand.yml` with Bootstrap customisation:

A code editor window with a dark theme. The title bar shows three colored circles (red, yellow, green) on the left and the filename "_quarto.yml" on the right. The editor contains a YAML configuration for Quarto. Line 1: "format:" in orange. Line 2: " html:" in orange. Line 3: " theme:" in orange. Line 4: " - cosmo" in cyan. Line 5: " - brand" in cyan. Line 6: " - styles.scss" in cyan. Line 7: "css: custom-components.css" in cyan. A clipboard icon is in the top right corner of the editor area. A circled number "1" is in the bottom right corner of the editor area.

```
1 format:
2   html:
3     theme:
4       - cosmo
5       - brand
6       - styles.scss
7   css: custom-components.css
```

Brand Integration

Combine `brand.yml` with Bootstrap customisation:



```
1 format:
2   html:
3     theme:
4       - cosmo
5       - brand
6       - styles.scss
7   css: custom-components.css
```

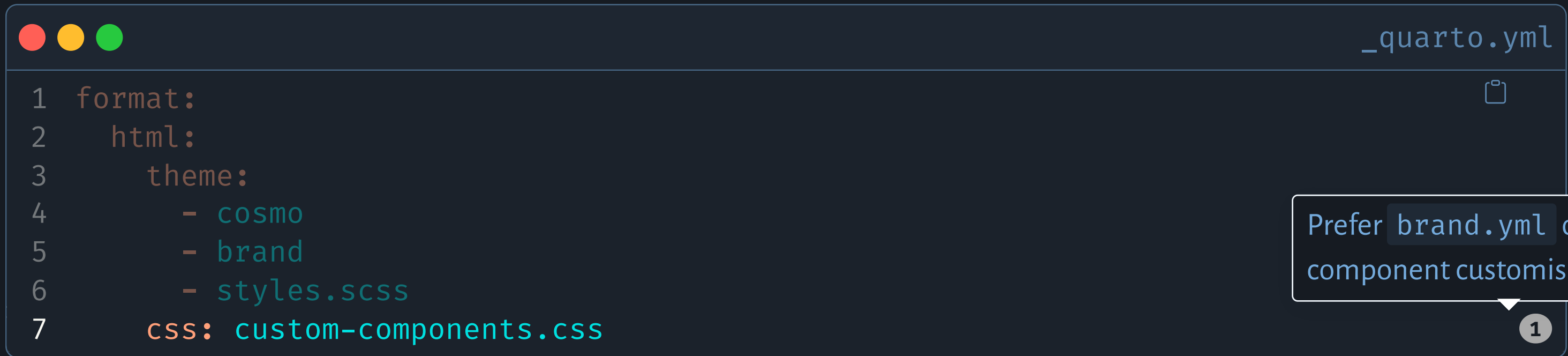
The code editor window displays the content of the `_quarto.yml` file. It features a dark theme with syntax highlighting. The file name `_quarto.yml` is shown in the top right corner. A clipboard icon is visible next to line 1, and a circled '1' icon is next to line 7.

Note

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

Brand Integration

Combine `brand.yml` with Bootstrap customisation:



```
1 format:
2   html:
3     theme:
4       - cosmo
5       - brand
6       - styles.scss
7   css: custom-components.css
```

The code editor window has a title bar with three colored circles (red, yellow, green) on the left and the filename `_quarto.yml` on the right. A clipboard icon is visible in the top right corner of the editor area. A small circular badge with the number '1' is located at the bottom right of the editor window.

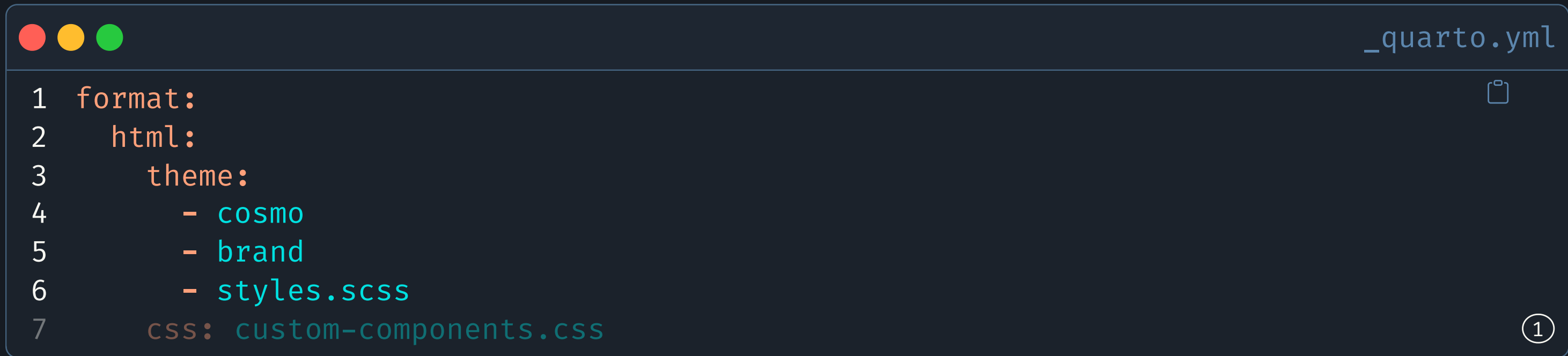
Prefer `brand.yml` or SCSS for component customisation.

Note

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

Brand Integration

Combine `brand.yml` with Bootstrap customisation:

A code editor window with a dark theme. The title bar shows three colored circles (red, yellow, green) on the left and the filename "_quarto.yml" on the right. The editor contains a YAML configuration for Bootstrap themes. Line numbers 1 through 7 are on the left. The code is: 1 format: 2 html: 3 theme: 4 - cosmo 5 - brand 6 - styles.scss 7 css: custom-components.css. There is a clipboard icon on the right side of the editor area and a circled number 1 at the bottom right corner of the editor window.

```
1 format:
2   html:
3     theme:
4       - cosmo
5       - brand
6       - styles.scss
7   css: custom-components.css
```

Note

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

- What Are Brand Extensions?
- Creating a Brand Extension

Brand Extensions Development

What Are Brand Extensions?

What Are Brand Extensions?

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

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:

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.

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.

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.

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.

Creating a Brand Extension

Creating a Brand Extension

Step 1: Scaffold Extension



BASH

```
1 quarto create extension brand my-brand  
2 cd my-brand
```



Creating a Brand Extension

Step 1: Scaffold Extension



BASH

```
1 quarto create extension brand my-brand
2 cd my-brand
```



This creates:

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

2 directories, 5 files

Creating a Brand Extension

Step 2: Configure Extension

● ● ●

Quarto

```
1 title: My-brand
2 author: First Last
3 version: 1.0.0
4 quarto-required: "≥ 1.9.0"
5 contributes:
6   metadata:
7     project:
8       brand: brand.yml
```

Creating a Brand Extension

Step 3: Design Brand System

```

_extensions/my-brand/brand.yml
1 meta:
2   name: "My Organisation"
3   link: "https://myorg.com"
4
5 color:
6   palette:
7     # Primary brand colours
8     brand-blue: "#2E86AB"
9     brand-coral: "#A23B72"
10    brand-gold: "#F4A261"
11    # Supporting colours
12    neutral-light: "#F8F9FA"
13    neutral-dark: "#343A40"
14    success-green: "#28A745"
15    warning-amber: "#FFC107"
16    # Semantic mappings
17    primary: brand-blue
18    secondary: brand-coral
19    success: success-green
20    warning: warning-amber
21    foreground: neutral-dark
22    background: neutral-light
23
24 logo:
25   small: assets/logo/favicon.png
26   medium: assets/logo/logo-horizontal.png
27   large: assets/logo/logo-vertical.png
28
29 typography:
30   fonts:
31     - family: "Source Sans Pro"
32       source: "google"
33       weight: [300, 400, 600, 700]
34
35   base: "Source Sans Pro"

```

```

_extensions/my-brand/brand.yml
1 defaults:
2   bootstrap:
3     defaults:
4       # Enhanced navigation
5       navbar-padding-y: 1rem
6       navbar-brand-font-size: 1.5rem
7
8       # Improved buttons
9       btn-padding-y: 0.5rem
10      btn-padding-x: 1.5rem
11      btn-font-weight: 600
12
13      # Better spacing
14      spacer: 1rem
15
16 rules: |
17   /* Professional enhancements */
18   .navbar-brand {
19     font-weight: 700;
20   }
21
22   .btn-primary {
23     border-radius: 2rem;
24     transition: all 0.3s ease;
25   }
26
27   .btn-primary:hover {
28     transform: translateY(-1px);
29   }
30
31   /* Content styling */
32   .title-block-header {
33     padding: 3rem 2rem;
34     margin-bottom: 2rem;
35     border-radius: 0.5rem;

```

Creating a Brand Extension

Step 4: Add Assets

Organise brand assets:



```
1 extensions/my-brand/  
2 |  _extension.yml  
3 |  brand.yml  
4 |  assets/  
5 |    logo/  
6 |      favicon.png  
7 |      logo-horizontal.png  
8 |      logo-vertical.png  
9 |    fonts/  
10 |      custom-font.woff2
```

The image shows a code editor window with a dark theme. The title bar has three colored circles (red, yellow, green) on the left and the text 'TXT' on the right. The editor area displays a file tree structure for brand assets. The root directory is 'extensions/my-brand/'. Inside this directory, there are three sub-items: '_extension.yml', 'brand.yml', and 'assets/'. The 'assets/' directory contains two sub-items: 'logo/' and 'fonts/'. The 'logo/' directory contains three files: 'favicon.png', 'logo-horizontal.png', and 'logo-vertical.png'. The 'fonts/' directory contains one file: 'custom-font.woff2'. The lines are numbered from 1 to 10 on the left side of the editor.

Creating a Brand Extension

Step 5: Create Example

```
example.qmd
1 ---
2 title: "Brand Extension Showcase"
3 subtitle: "Demonstrating consistent branding"
4 author: "Your Name"
5 format:
6   html:
7     theme:
8       - quartz
9       - brand
10 ---
11
12 # Welcome to Our Brand System
13
14 This document demonstrates the power of brand extensions in Quarto.
15
16 [Primary Action]{.btn .btn-primary}
17 [Secondary Action]{.btn .btn-secondary}
```

! Important

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

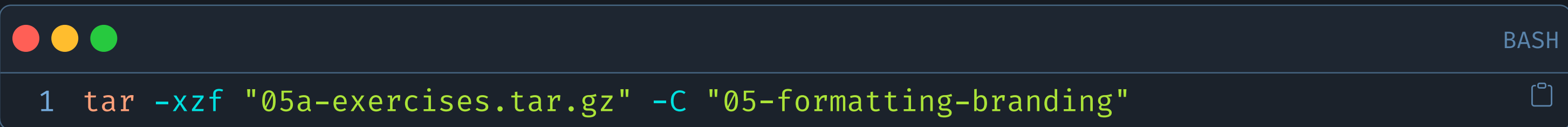
- Exercise Overview
- Part 1: Brand Extension Setup
- Part 2: Testing
- Success Criteria

Hands-On Exercise: Building Your Brand Extension

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](#)

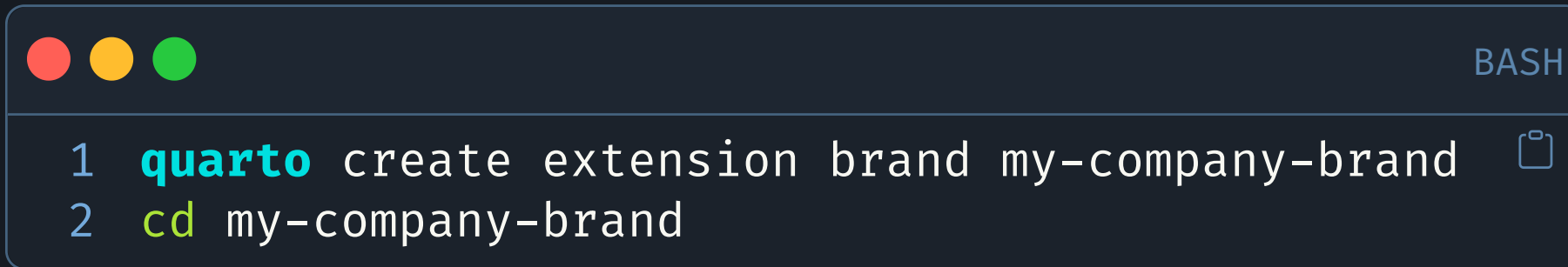


```
1 tar -xzf "05a-exercises.tar.gz" -C "05-formatting-branding"
```

BASH 

Part 1: Brand Extension Setup

1. Generate extension scaffold:



```
1 quarto create extension brand my-company-brand
2 cd my-company-brand
```

2. Define your brand identity.

3. Add styling.

Part 2: Testing

1. Build test project:

```
1 quarto create project
```

2. Test across formats:

```
1 # Test HTML output  
2 quarto render example.qmd --to html  
3  
4 # Test presentation  
5 quarto render example.qmd --to revealjs  
6  
7 # Test Typst output  
8 quarto render example.qmd --to typst
```

3. Verify brand consistency.

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.

- Understanding Pandoc Templates
- Basic Variable Syntax
- Conditional Content
- Loops for Multiple Items
- Separator Usage

Pandoc Templating System

Understanding Pandoc Templates

Understanding Pandoc Templates

Quarto uses Pandoc's templating system to generate output.

Understanding Pandoc Templates

Quarto uses Pandoc's templating system to generate output.

Key concepts:

Understanding Pandoc Templates

Quarto uses Pandoc's templating system to generate output.

Key concepts:

- **Variables** - `$variable$` inserts YAML metadata.

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.

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.

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.

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.

Basic Variable Syntax

Basic Variable Syntax

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

Basic Variable Syntax

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

● ● ●

Quarto Document

```
1 ---
2 title: "My Report"
3 author: "Jane Smith"
4 date: "2025-01-15"
5 ---
```

● ● ●

Template Input

```
1 Title: $title$
2 Author: $author$
3 Date: $date$
```

● ● ●

Template Output

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

Basic Variable Syntax

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

Quarto Document

```
1 ---
2 title: "My Report"
3 author: "Jane Smith"
4 date: "2025-01-15"
5 ---
```

Template Input

```
1 Title: $title$
2 Author: $author$
3 Date: $date$
```

Template Output

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



Tip

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

Basic Variable Syntax

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

Quarto Document

```
1 ---  
2 title: "My Report"  
3 author: "Jane Smith"  
4 date: "2025-01-15"  
5 ---
```

Template Input

```
1 Title: $title$  
2 Author: $author$  
3 Date: $date$
```

Template Output

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



Tip

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

See [Article Templates](#) for templates and partials.

Conditional Content

Conditional Content

Show content only when a variable exists:

Conditional Content

Show content only when a variable exists:

Quarto Document

```
1 ---
2 title: "Report"
3 subtitle: "Q4 Analysis"
4 # No author specified
5 ---
```

Template Input

```
1 Title: $title$
2
3 $if(subtitle)$
4 Subtitle: $subtitle$
5 $endif$
6
7 $if(author)$
8 Author: $author$
9 $endif$
```

Template Output

```
1 Title: Report
2 Subtitle: Q4 Analysis
```

Conditional Content

Show content only when a variable exists:

Quarto Document

```
1 ---
2 title: "Report"
3 subtitle: "Q4 Analysis"
4 # No author specified
5 ---
```

Note

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

Template Input

```
1 Title: $title$
2
3 $if(subtitle)$
4 Subtitle: $subtitle$
5 $endif$
6
7 $if(author)$
8 Author: $author$
9 $endif$
```

Template Output

```
1 Title: Report
2 Subtitle: Q4 Analysis
```


Conditional Content

Show content only when a variable exists:

Quarto Document

```

1 ---
2 title: "Report"
3 subtitle: "Q4 Analysis"
4 # No author specified
5 ---

```

Note

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

Template Input

```

1 Title: $title$
2
3 $if(subtitle)$
4 Subtitle: $subtitle$
5 $endif$
6
7 $if(author)$
8 Author: $author$
9 $endif$

```

Template Output

```

1 Title: Report
2 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.

Loops for Multiple Items

Loops for Multiple Items

Process lists from YAML metadata:

Loops for Multiple Items

Process lists from YAML metadata:

```
1 ---
2 authors:
3   - name: Jane Smith
4     email: jane@example.com
5   - name: John Doe
6     email: john@example.com
7 ---
```

```
1 Authors:
2 $for(authors)$
3 - $it.name$ ($it.email$)
4 $endfor$
```

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

Loops for Multiple Items

Process lists from YAML metadata:

```
1 ---
2 authors:
3   - name: Jane Smith
4     email: jane@example.com
5   - name: John Doe
6     email: john@example.com
7 ---
```

```
1 Authors:
2 $for(authors)$
3 - $it.name$ ($it.email$)
4 $endfor$
```

```
1 Authors:
2 - Jane Smith (jane@example.com)
3 - 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.

Separator Usage

Add separators between items:

Quarto Document

1

2

tags:

3

- Research

4

- Data Science

5

- Machine Learning

6

Template Input

1

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

Template Output

1

Tags: Research, Data Science, Machine Learning

Template Input

1

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

Template Output

1

Tags: ResearchData ScienceMachine Learning

- Exercise Overview
- Part 1: Template Modification
- Part 2: Customise Further
- Success Criteria

Hands-On Exercise: Pandoc Templating

Exercise Overview

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

Example Code:  [Exercises Partials](#)



BASH

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



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.

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.

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.

- Typst in Quarto
- Typst Template Structure
- Simple Customisation Example
- Creating a Typst Format

Typst Customisation (For the Curious)

Typst in Quarto

Typst in Quarto

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

Typst in Quarto

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

Why Typst?

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.

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.

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.

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.

Typst Template Structure

Typst Template Structure

Quarto's Typst templates use two key files:

Typst Template Structure

Quarto's Typst templates use two key files:

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

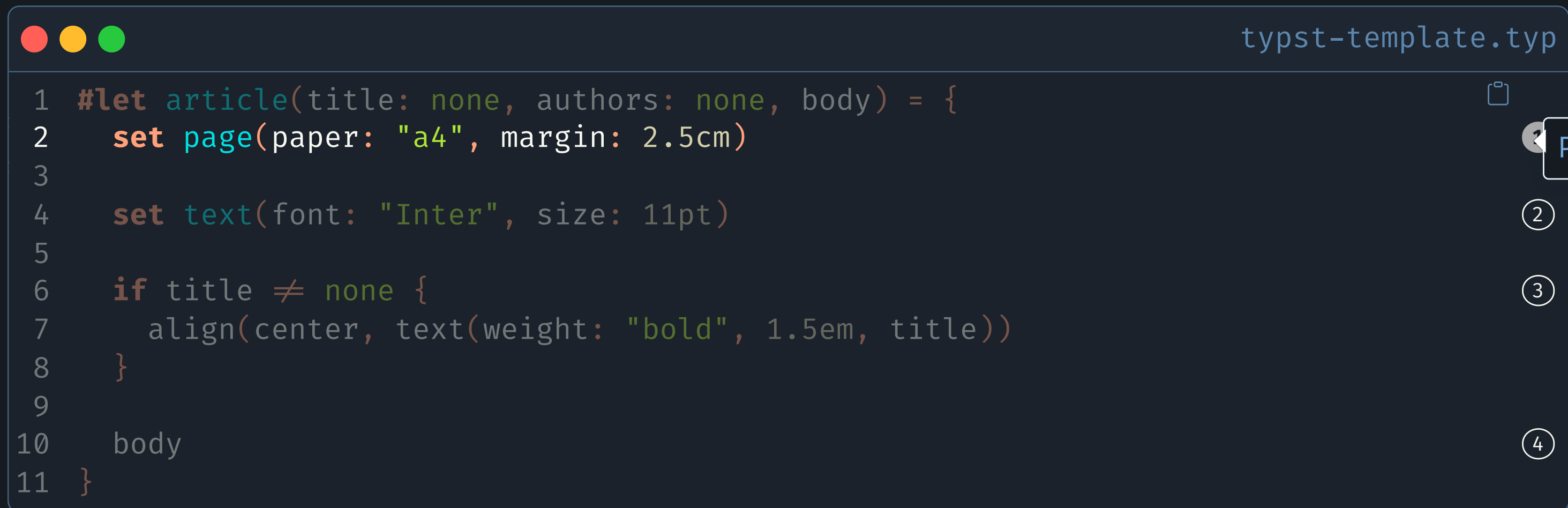
```
typst-template.typ

1 #let article(title: none, authors: none, body) = {
2   set page(paper: "a4", margin: 2.5cm)
3
4   set text(font: "Inter", size: 11pt)
5
6   if title ≠ none {
7     align(center, text(weight: "bold", 1.5em, title))
8   }
9
10  body
11 }
```

Typst Template Structure

Quarto's Typst templates use two key files:

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



```
1 #let article(title: none, authors: none, body) = {
2   set page(paper: "a4", margin: 2.5cm)
3
4   set text(font: "Inter", size: 11pt)
5
6   if title != none {
7     align(center, text(weight: "bold", 1.5em, title))
8   }
9
10  body
11 }
```

Page setup.

Typst Template Structure

Quarto's Typst templates use two key files:

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



```
1 #let article(title: none, authors: none, body) = {
2   set page(paper: "a4", margin: 2.5cm)
3
4   set text(font: "Inter", size: 11pt)
5
6   if title ≠ none {
7     align(center, text(weight: "bold", 1.5em, title))
8   }
9
10  body
11 }
```

Typography settings.

Typst Template Structure

Quarto's Typst templates use two key files:

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



```
1 #let article(title: none, authors: none, body) = {
2   set page(paper: "a4", margin: 2.5cm)
3
4   set text(font: "Inter", size: 11pt)
5
6   if title ≠ none {
7     align(center, text(weight: "bold", 1.5em, title))
8   }
9
10  body
11 }
```

①

②

③ Conditional title block

④

Typst Template Structure

Quarto's Typst templates use two key files:

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



```
1 #let article(title: none, authors: none, body) = {
2   set page(paper: "a4", margin: 2.5cm)
3
4   set text(font: "Inter", size: 11pt)
5
6   if title ≠ none {
7     align(center, text(weight: "bold", 1.5em, title))
8   }
9
10  body
11 }
```

typst-template.typ

1

2

3

4

5

6

7

8

9

10

11

Main content insertion point

Typst Template Structure

Typst Template Structure

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

```
typst-show.typ

1 #show: doc ⇒ article(
2   $if(title)$
3   title: [$title$],
4   $endif$
5   $if(by-author)$
6   authors: [$for(by-author)$it.name.literal$$sep$, $endfor$],
7   $endif$
8   doc,
9 )
```

Typst Template Structure

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

```
typst-show.typ

1 #show: doc ⇒ article(
2   $if(title)$
3   title: [$title$],
4   $endif$
5   $if(by-author)$
6   authors: [$for(by-author)$it.name.literal$$sep$, $endfor$],
7   $endif$
8   doc,
9 )
```

i Note

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

Typst Template Structure

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

```
1 #show: doc ⇒ article(  
2   $if(title)$  
3   title: [$title$],  
4   $endif$  
5   $if(by-author)$  
6   authors: [$for(by-author)$it.name.literal$$sep$, $endfor$],  
7   $endif$  
8   doc,  
9 )
```

typst-show.typ

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

1

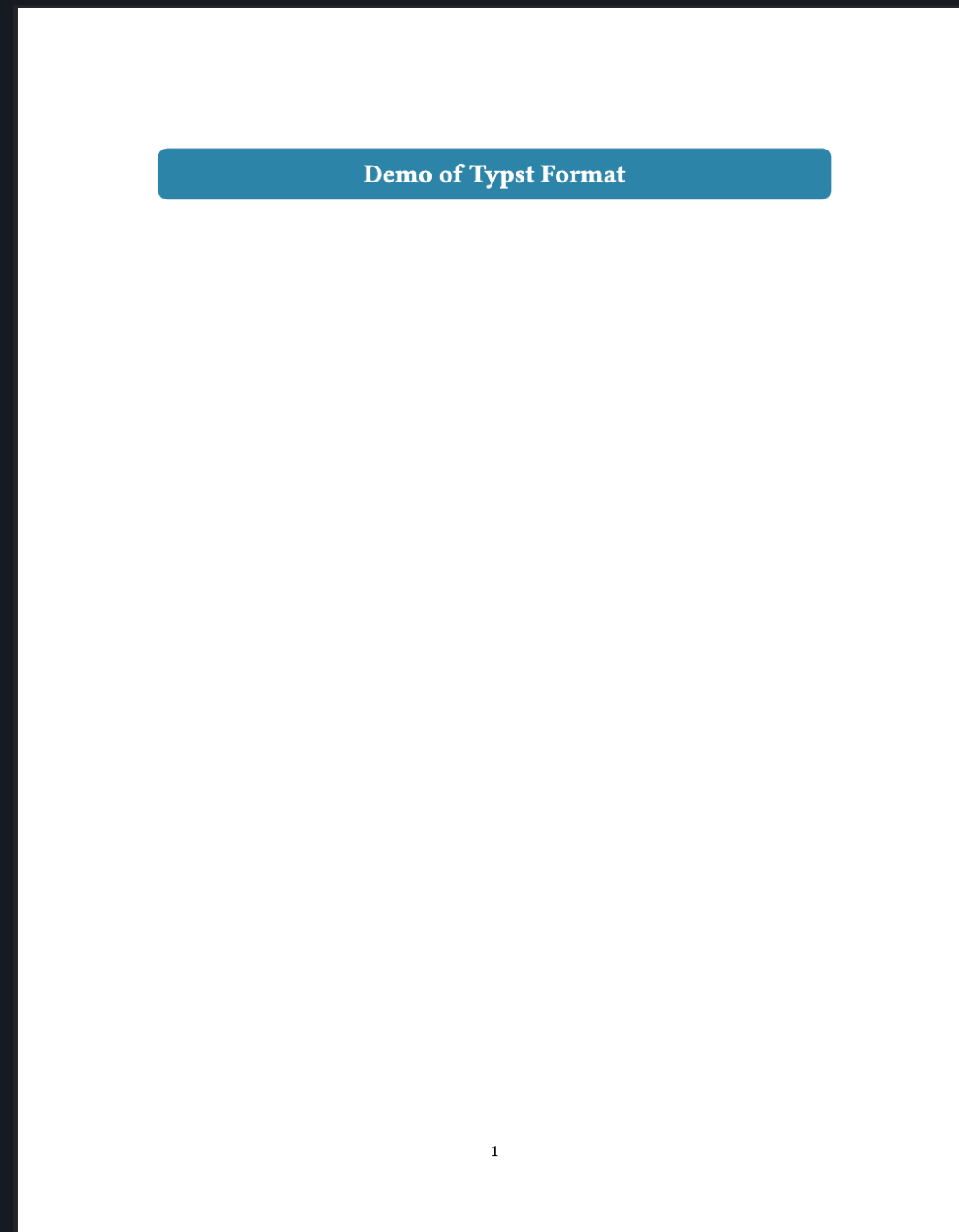
i Note

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

Simple Customisation Example

```
typst-template.typ
1 #let article(
2   title: none,
3   body
4 ) = {
5   let primary = rgb("#2E86AB")
6
7   if title ≠ none {
8     block(
9       fill: primary,
10      inset: 1em,
11      radius: 0.5em,
12      width: 100%
13    )[
14      #align(center)[
15        #text(
16          fill: white,
17          weight: "bold",
18          size: 1.5em,
19          title
20        )
21      ]
22    ]
23  }
24
25  body
26 }
```

```
typst-show.typ
1 #show: doc ⇒ article(
2   $if(title)$
3   title: [$title$],
4   $endif$
5   doc,
6 )
```



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

Creating a Typst Format

Creating a Typst Format

Step 1: Scaffold Extension



BASH

```
1 quarto create extension format:typst my-format  
2 cd my-format
```



Creating a Typst Format

Step 1: Scaffold Extension

BASH

```
1 quarto create extension format:typst my-format
2 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
```

Creating a Typst Format

Step 1: Scaffold Extension

```
1 quarto create extension format:typst my-format
2 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.

Creating a Typst Format

Step 2: Integration with `brand.yml`

```

1 color:
2   primary: "#2E86AB"
3
4 logo:
5   medium: "assets/logo.png"
  
```

```

1 #show: doc => article(
2   $if(title)$
3   title: [$title$],
4   title_colour: brand-color.primary,
5   $endif$
6   doc,
7 )
  
```

```

1 $if(logo)$
2 #set page(
3   header: context {
4     if counter(page).get().first() = 1{
5       place(top + left, dy: 0.5cm, image("$logo.path$", width: 2cm))
6     }
7   }
8 )
9 $endif$
  
```

```

1 #let article(
2   title: none,
3   title_colour: "#000000",
4   body
5 ) = {
6   set text(size: 11pt)
7   if title != none {
8     align(center)[
9       #text(
10        fill: rgb(title_colour),
11        weight: "bold",
12        size: 1.5em,
13        title
14      )
15    ]
16  }
17   body
18 }
  
```

- Exercise Overview
- Part 1: Format Extension Setup
- Part 2: Customise and Test Your Format
- Part 3: Optional HTML Variant
- Success Criteria

Hands-On Exercise: Creating a Format Extension

Exercise Overview

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

Example Code:  [Exercices Typst](#)



BASH

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



Part 1: Format Extension Setup

1. Generate format extension scaffold:

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

BASH

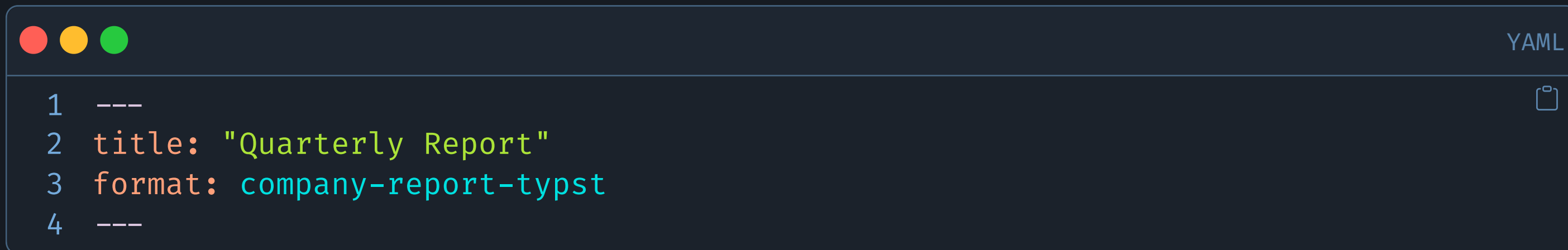
2. Add **brand.yml** and configure extension in **_extension.yml**:

```
1 contributes:
2   formats:
3     typst:
4       template-partials:
5         - assets-typst/typst-template.typ
6         - assets-typst/typst-show.typ
7         - assets-typst/page.typ
8   metadata:
9     project:
10      brand: brand.yml
```

YAML

Part 2: Customise and Test Your Format

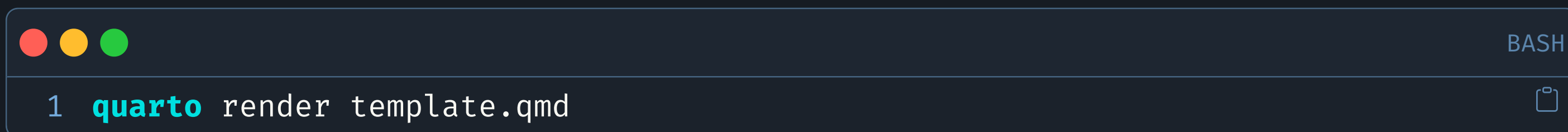
1. Define your template in `typst-template.typ` and `typst-show.typ`.
2. Use your format in `template.qmd`:



```
1 ---
2 title: "Quarterly Report"
3 format: company-report-typst
4 ---
```

A terminal window with a dark background and three colored window control buttons (red, yellow, green) in the top-left corner. The title bar on the right says "YAML". The terminal content shows a YAML file with four lines: a separator line, a title, a format, and another separator line.

3. Render and verify:



```
1 quarto render template.qmd
```

A terminal window with a dark background and three colored window control buttons (red, yellow, green) in the top-left corner. The title bar on the right says "BASH". The terminal content shows a single command: `quarto render template.qmd`.

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

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



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.



Mickaël CANOUIL, *Ph.D.*

Independent Contractor

pro@mickael.canouil.dev



Website

mickael.canouil.fr



GitHub

[@mcanouil](https://github.com/mcanouil)



LinkedIn

[MickaelCanouil](https://www.linkedin.com/company/MickaelCanouil)



Bluesky

[@mickael.canouil.fr](https://bsky.app/profile/@mickael.canouil.fr)



Mastodon

[@MickaelCanouil](https://mastodon.social/@MickaelCanouil)