

Formats & Projects

Session 4

Mickaël CANOUIL, *Ph.D.* 

pro@mickael.canouil.dev

Independent Contractor

Saturday, the 7th of February, 2026

- Session 4: Formats & Projects
- Session Objectives
- Learning Objectives

Session Overview

Session 4: Formats & Projects

- **Quarto Projects**

- Project types and architecture: default, website, blog, manuscript, book.
- Project type decision framework based on publishing goals.
- Understanding project structure and configuration files.

- **Advanced Project Features**

- Shared metadata and configuration with `_metadata.yml`.
- Pre and post-render scripts for workflow automation.
- Project profiles for different deployment scenarios.
- Environment variables and resource management.

- **Multiple Formats**

- Multi-format output optimisation with format-specific options.
- Conditional content by format using content-visible/hidden classes.
- Cross-format resource management and figure optimisation.

Session Objectives

This session explores Quarto's project architecture and format ecosystem, covering project types, multi-format optimisation, and template systems for scalable workflows.

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:

- Select appropriate project types for different publishing scenarios.

Learning Objectives

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

- Select appropriate project types for different publishing scenarios.
- Configure multi-format outputs with format-specific optimisations.

Learning Objectives

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

- Select appropriate project types for different publishing scenarios.
- Configure multi-format outputs with format-specific optimisations.
- Use template systems for rapid project initiation.

Learning Objectives

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

- Select appropriate project types for different publishing scenarios.
- Configure multi-format outputs with format-specific optimisations.
- Use template systems for rapid project initiation.
- Implement collaborative workflows with proper project structure.

Learning Objectives

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

- Select appropriate project types for different publishing scenarios.
- Configure multi-format outputs with format-specific optimisations.
- Use template systems for rapid project initiation.
- Implement collaborative workflows with proper project structure.
- Understand format capabilities and limitations.

- Project Types and Architecture
- Project Type Decision Framework
- Default Project Structure
- Website Projects
- Blog Projects (website + listing)
- Manuscript Projects
- Book Projects

Quarto Projects

Project Types and Architecture

Project Types and Architecture

Recall the project types introduced in Session 1. Here we examine their configuration and architecture in more detail.

Project Types and Architecture

Recall the project types introduced in Session 1. Here we examine their configuration and architecture in more detail.



Bash

```
1 quarto create project
```



Project Types and Architecture

Recall the project types introduced in Session 1. Here we examine their configuration and architecture in more detail.

Bash

1 **quarto** create project

BASH

1 **? Type**
2 **> default**
3 **website**
4 **blog**
5 **manuscript**
6 **book**
7 **confluence**

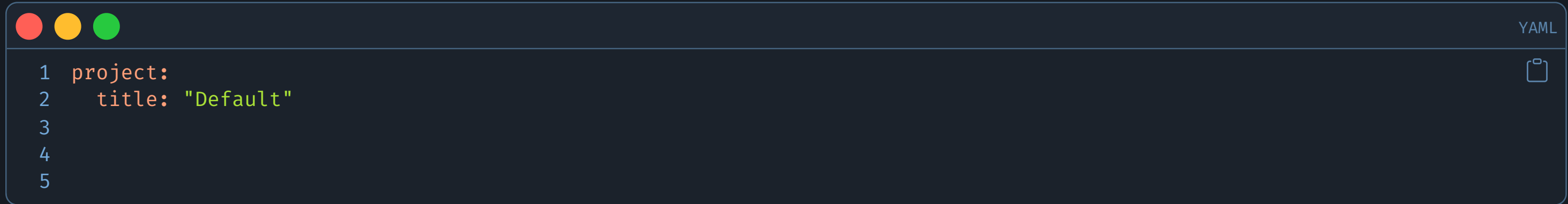
Project Type Decision Framework

Choose your project type based on your publishing goals:

Scenario	Project Type	Key Features
Multiple unrelated documents	default	Flexible structure, shared metadata
Single research paper	manuscript	Academic formatting, citations, submission-ready
Public documentation/blog	website / blog	Navigation, search, responsive design
Long-form content	book	Cross-chapter references, multiple outputs

Default Project Structure

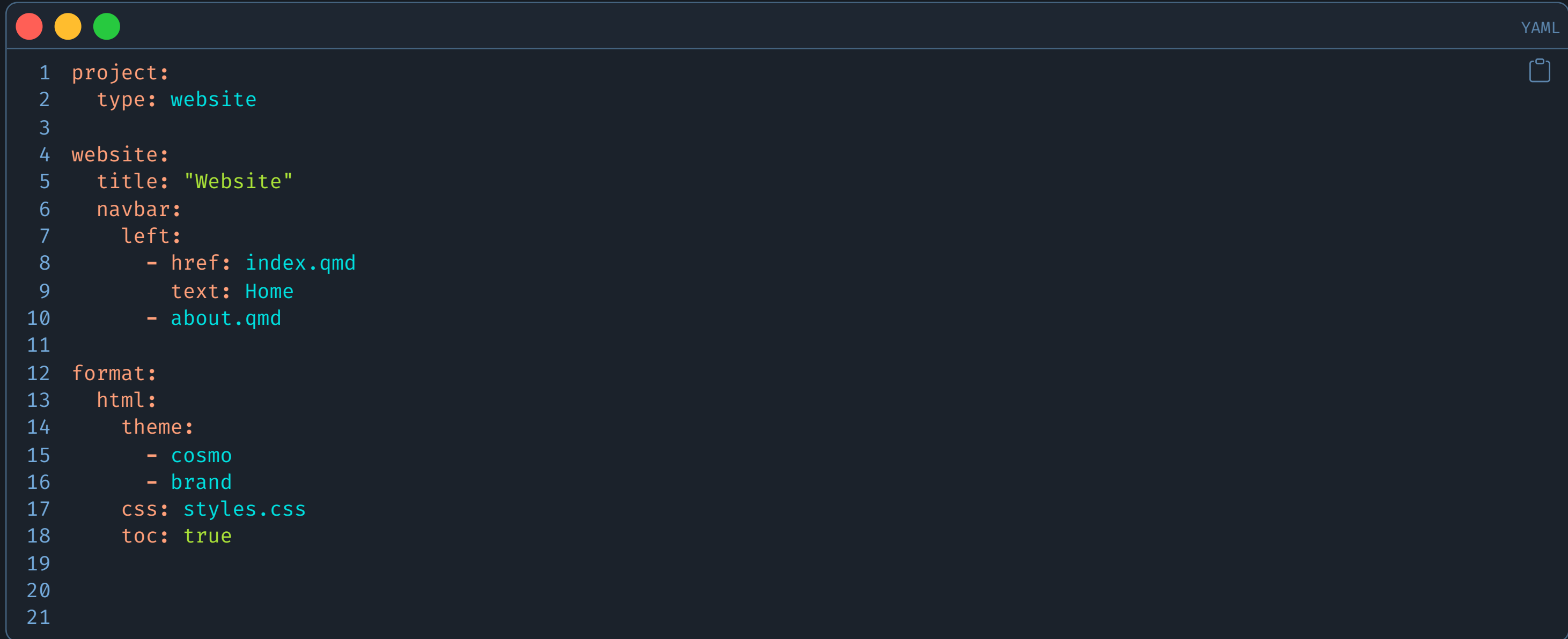
The most flexible option for diverse content:



```
1 project:
2   title: "Default"
3
4
5
```

Website Projects

Optimised for web publishing:



```
1 project:
2   type: website
3
4 website:
5   title: "Website"
6   navbar:
7     left:
8       - href: index.qmd
9         text: Home
10      - about.qmd
11
12 format:
13   html:
14     theme:
15       - cosmo
16       - brand
17     css: styles.css
18     toc: true
19
20
21
```

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.

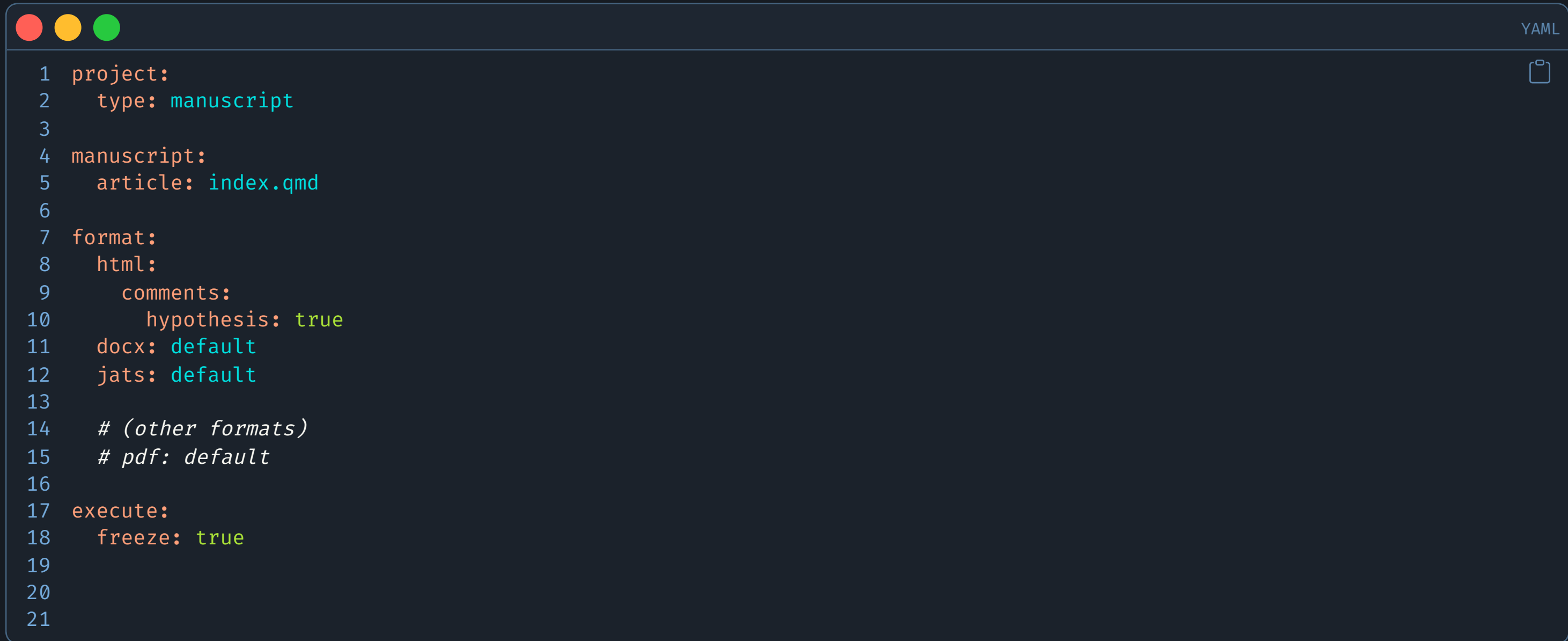
Blog Projects (website + listing)

Designed for regular posts and updates:

```
1 project:
2   type: website
3
4 website:
5   description: "A blog built with Quarto"
6   site-url: https://your-website-url.example.com # you must change this appropriately for RSS feeds to work properly
7   title: "Blog"
8   navbar:
9     right:
10      - about.qmd
11      - icon: github
12        href: https://github.com/
13      - icon: bluesky
14        href: https://bsky.app/
15 format:
16   html:
17     theme:
18      - cosmo
19      - brand
20     css: styles.css
21
22
```

Manuscript Projects

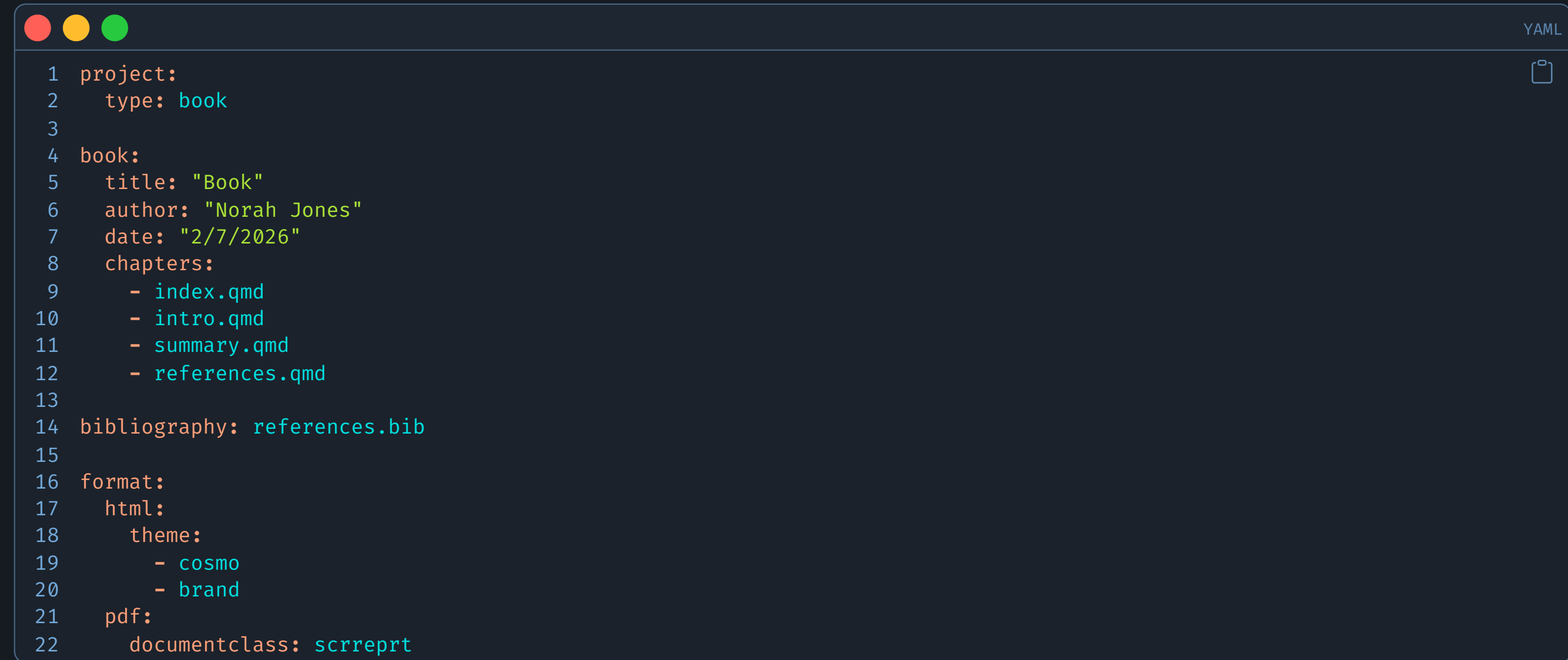
Optimised for academic publishing:



```
1 project:
2   type: manuscript
3
4 manuscript:
5   article: index.qmd
6
7 format:
8   html:
9     comments:
10      hypothesis: true
11   docx: default
12   jats: default
13
14   # (other formats)
15   # pdf: default
16
17 execute:
18   freeze: true
19
20
21
```

Book Projects

For comprehensive, structured content:



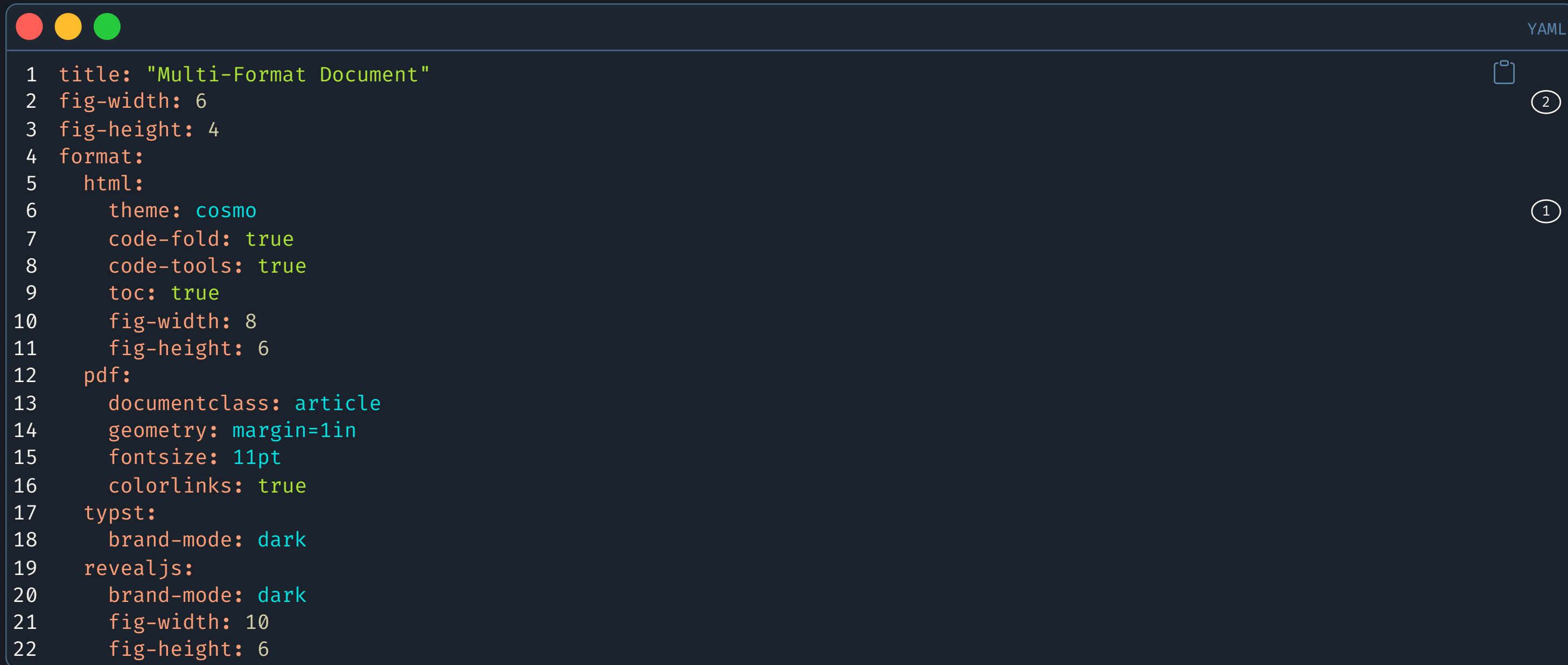
```
1 project:
2   type: book
3
4 book:
5   title: "Book"
6   author: "Norah Jones"
7   date: "2/7/2026"
8   chapters:
9     - index.qmd
10    - intro.qmd
11    - summary.qmd
12    - references.qmd
13
14 bibliography: references.bib
15
16 format:
17   html:
18     theme:
19       - cosmo
20       - brand
21   pdf:
22     documentclass: scrreprt
```

- Multi-Format Output Optimisation
- Conditional Content by Format
- Cross-Format Resource Management

Multiple Formats

Multi-Format Output Optimisation

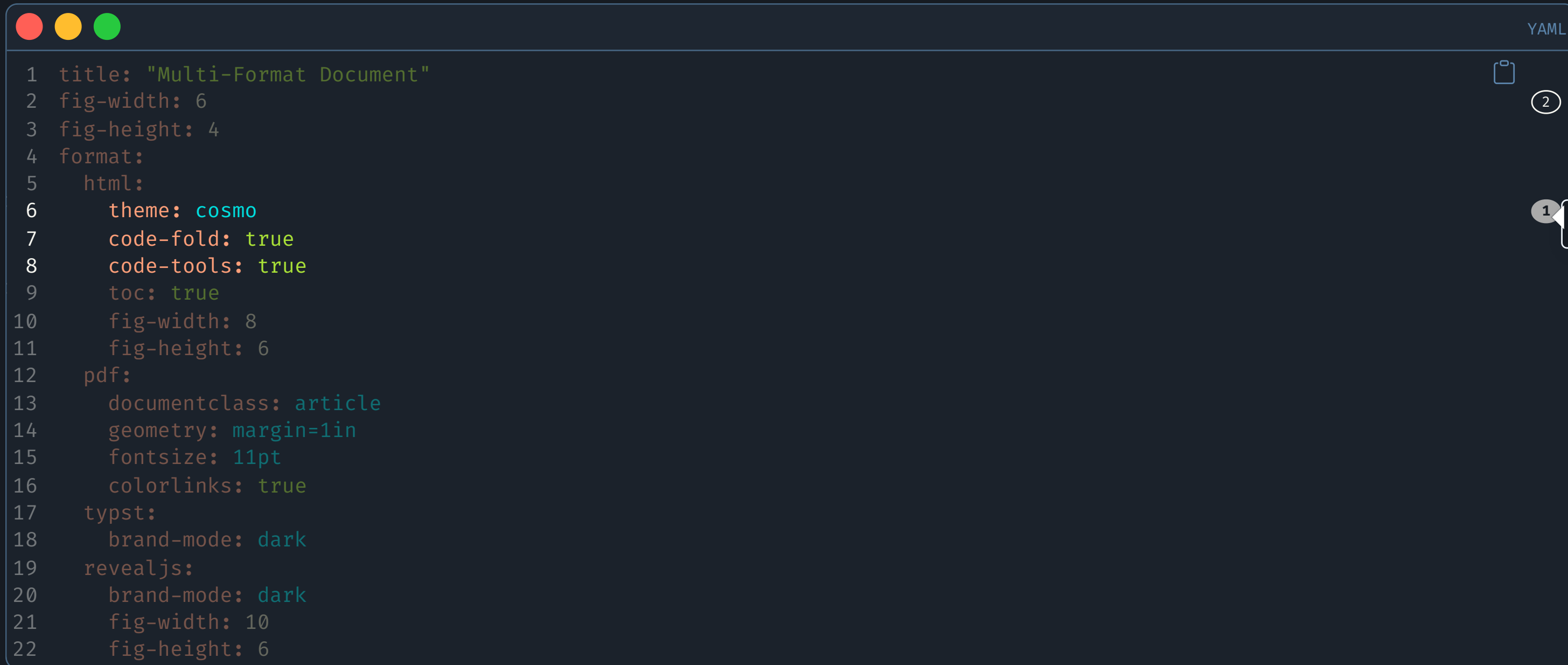
Balance capability with compatibility by configuring format-specific options that leverage each output type's strengths.



```
1 title: "Multi-Format Document"
2 fig-width: 6
3 fig-height: 4
4 format:
5   html:
6     theme: cosmo
7     code-fold: true
8     code-tools: true
9     toc: true
10    fig-width: 8
11    fig-height: 6
12  pdf:
13    documentclass: article
14    geometry: margin=1in
15    fontsize: 11pt
16    colorlinks: true
17  typst:
18    brand-mode: dark
19  revealjs:
20    brand-mode: dark
21    fig-width: 10
22    fig-height: 6
```

Multi-Format Output Optimisation

Balance capability with compatibility by configuring format-specific options that leverage each output type's strengths.

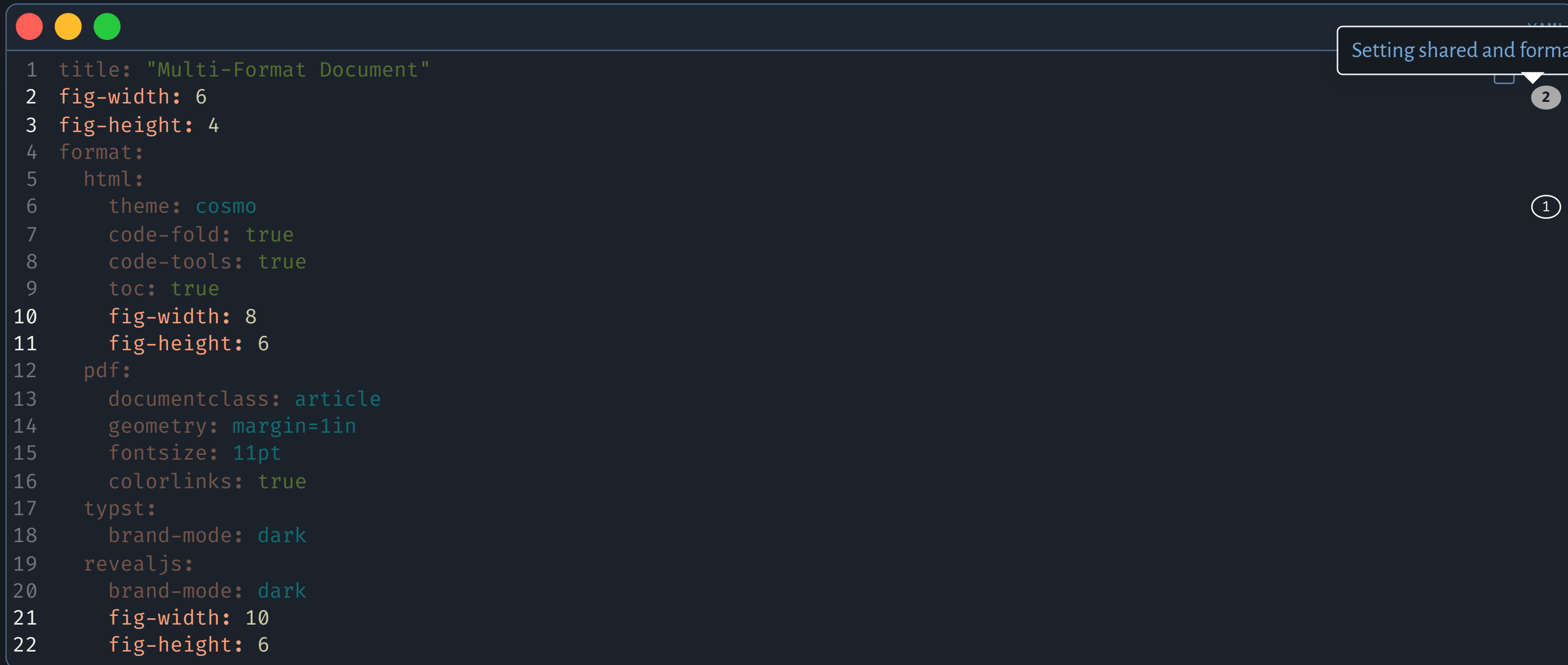


```
1 title: "Multi-Format Document"
2 fig-width: 6
3 fig-height: 4
4 format:
5   html:
6     theme: cosmo
7     code-fold: true
8     code-tools: true
9     toc: true
10    fig-width: 8
11    fig-height: 6
12  pdf:
13    documentclass: article
14    geometry: margin=1in
15    fontsize: 11pt
16    colorlinks: true
17  typst:
18    brand-mode: dark
19  revealjs:
20    brand-mode: dark
21    fig-width: 10
22    fig-height: 6
```

1 Setting format-specific options

Multi-Format Output Optimisation

Balance capability with compatibility by configuring format-specific options that leverage each output type's strengths.



```
1 title: "Multi-Format Document"
2 fig-width: 6
3 fig-height: 4
4 format:
5   html:
6     theme: cosmo
7     code-fold: true
8     code-tools: true
9     toc: true
10    fig-width: 8
11    fig-height: 6
12  pdf:
13    documentclass: article
14    geometry: margin=1in
15    fontsize: 11pt
16    colorlinks: true
17  typst:
18    brand-mode: dark
19  revealjs:
20    brand-mode: dark
21    fig-width: 10
22    fig-height: 6
```

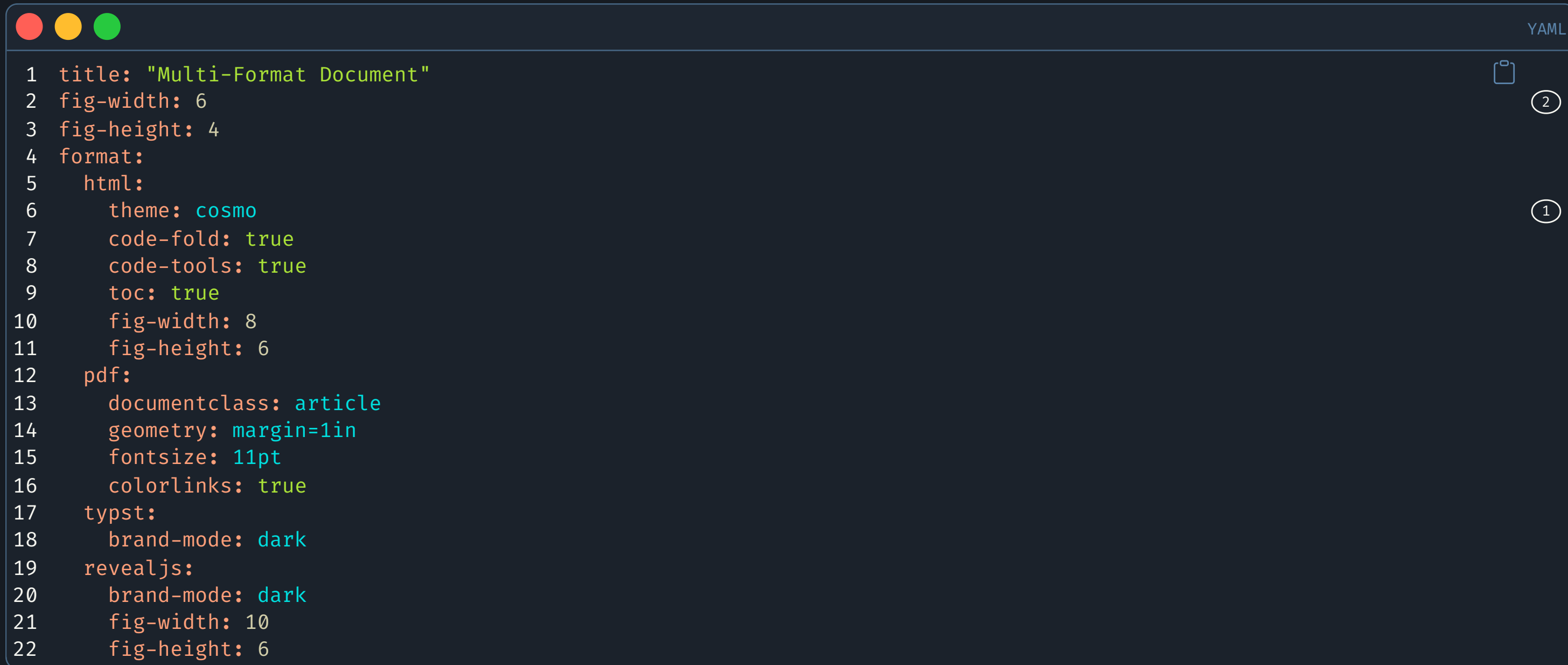
Setting shared and format-specific options.

1

2

Multi-Format Output Optimisation

Balance capability with compatibility by configuring format-specific options that leverage each output type's strengths.



```
1 title: "Multi-Format Document"
2 fig-width: 6
3 fig-height: 4
4 format:
5   html:
6     theme: cosmo
7     code-fold: true
8     code-tools: true
9     toc: true
10    fig-width: 8
11    fig-height: 6
12  pdf:
13    documentclass: article
14    geometry: margin=1in
15    fontsize: 11pt
16    colorlinks: true
17  typst:
18    brand-mode: dark
19  revealjs:
20    brand-mode: dark
21    fig-width: 10
22    fig-height: 6
```

Conditional Content by Format

Conditional Content by Format

Control what content appears in different output formats.

Conditional Content by Format

Control what content appears in different output formats.

Key Classes

- `.content-visible` - Show content only for specific formats.
- `.content-hidden` - Hide content from specific formats.

Conditional Content by Format

Control what content appears in different output formats.

Key Classes

- `.content-visible` - Show content only for specific formats.
- `.content-hidden` - Hide content from specific formats.

Conditions

- `when-format` / `unless-format` - Control visibility based on output format.
- `when-meta` / `unless-meta` - Control visibility based on metadata.
- `when-profile` / `unless-profile` - Control visibility based on profile.

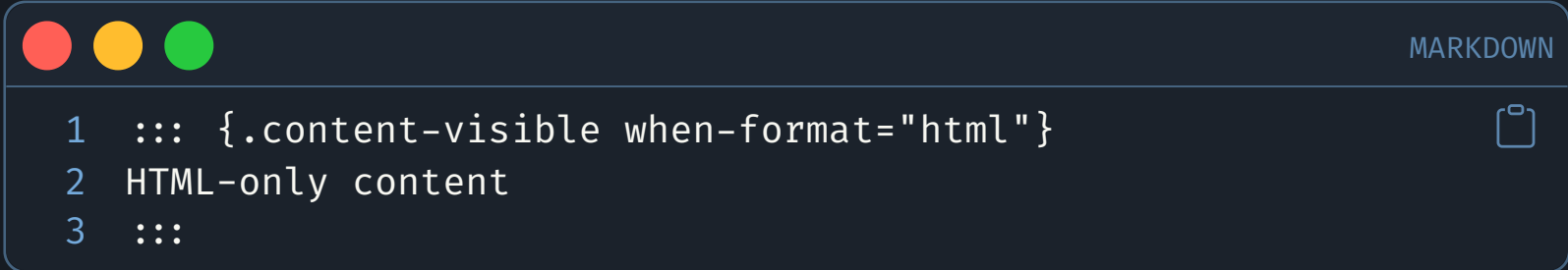
Conditional Content by Format

Conditional Content by Format

Syntax

Conditional Content by Format

Syntax

- 

```
1 ::: {.content-visible when-format="html"}
2 HTML-only content
3 :::
```

Conditional Content by Format

Syntax

-  MARKDOWN
1 ::: {.content-visible when-format="html"}
2 HTML-only content
3 :::
-  MARKDOWN
1 [Typst/PDF only text]{.content-visible when-format="typst"} 

Conditional Content by Format

Syntax

-  MARKDOWN
1 ::: {.content-visible when-format="html"}
2 HTML-only content
3 :::
-  MARKDOWN
1 *[Typst/PDF only text]*{.content-visible when-format="typst"} 
-  MARKDOWN
1 ```{.python .content-visible when-format="html"}
2 print("Only shows in HTML")
3 ```

Conditional Content by Format

Syntax

- 
MARKDOWN

```
1 ::: {.content-visible when-format="html"}
2 HTML-only content
3 :::
```
- 
MARKDOWN

```
1 [Typst/PDF only text]{.content-visible when-format="typst"}
```
- 
MARKDOWN

```
1 ```{.python .content-visible when-format="html"}
2 print("Only shows in HTML")
3 ```
```
- 
MARKDOWN

```
1 ::: {.content-visible when-format="html"}
2 ```{python}
3 print("Only shows in HTML but is evaluated in all formats")
4 ```
5 :::
```

Cross-Format Resource Management

Cross-Format Resource Management

Optimise images and assets for different outputs:

Cross-Format Resource Management

Optimise images and assets for different outputs:

● ● ●

YAML

1

format:

2

html:

3

fig-format: **svg** *# Vector graphics for web*

4

fig-dpi: 150

5

pdf:

6

fig-format: **pdf** *# Vector graphics for print*

7

fig-dpi: 300

8

typst:

9

fig-format: **png** *# Raster for compatibility*

10

fig-dpi: 150

- Shared Metadata and Configuration
- Resource Management
- Pre and Post Render Scripts
- Environment Variables
- Variables
- Project Profiles

Advanced Project Features

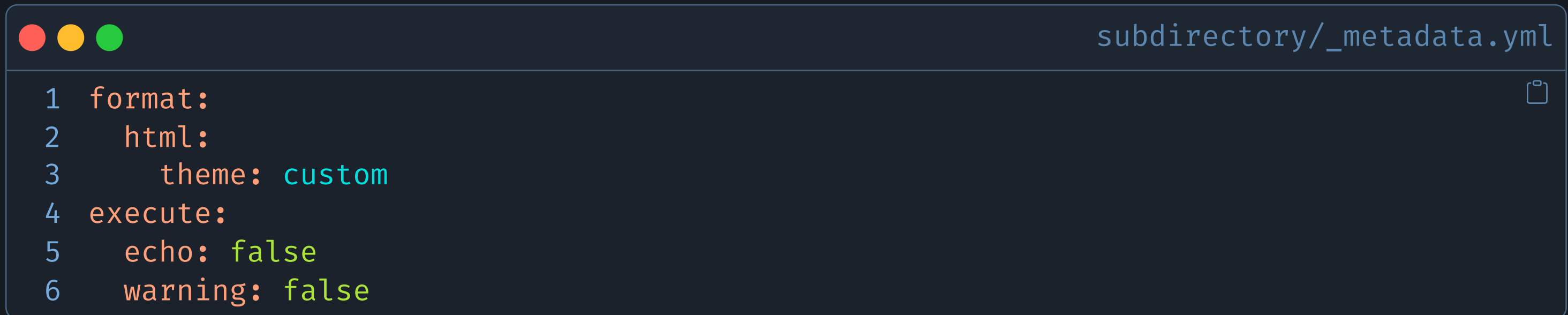
Shared Metadata and Configuration

Shared Metadata and Configuration

Use `_metadata.yml` for common settings to be shared by documents in a directory:

Shared Metadata and Configuration

Use `_metadata.yml` for common settings to be shared by documents in a directory:



```
subdirectory/_metadata.yml

1 format:
2   html:
3     theme: custom
4 execute:
5   echo: false
6   warning: false
```

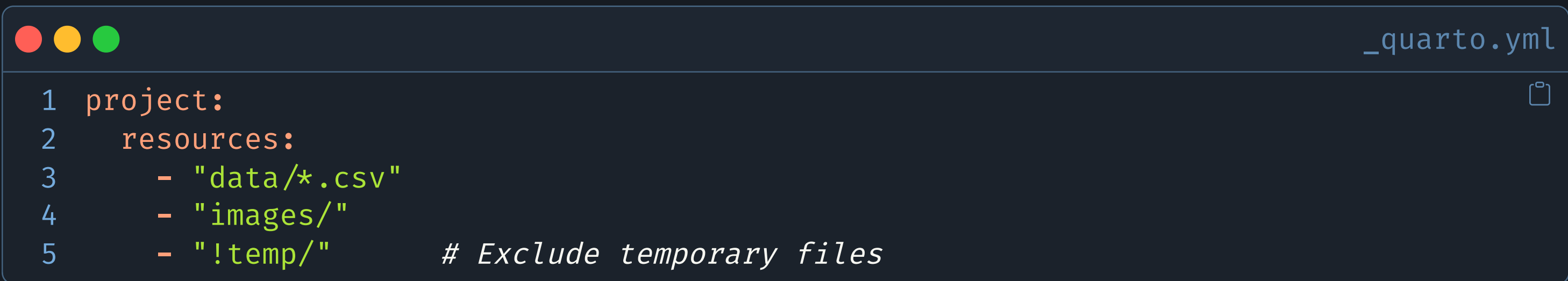
Resource Management

Resource Management

Organise assets that needs to be copied to the output directory:

Resource Management

Organise assets that needs to be copied to the output directory:



```
1 project:
2   resources:
3     - "data/*.csv"
4     - "images/"
5     - "!temp/"      # Exclude temporary files
```

_quarto.yml

Pre and Post Render Scripts

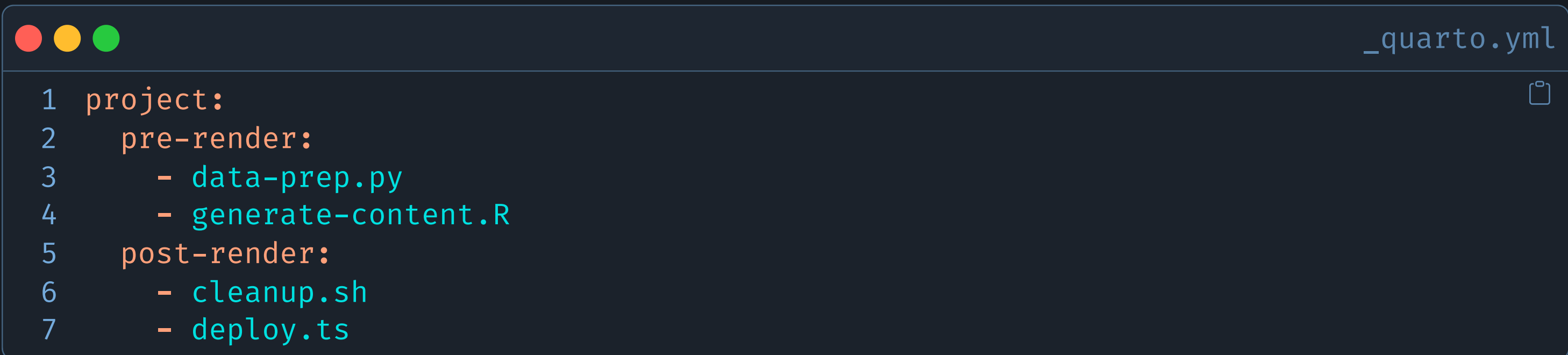
Pre and Post Render Scripts

Automate workflows before and after rendering.

Pre and Post Render Scripts

Automate workflows before and after rendering.

Configuration



```
1 project:
2   pre-render:
3     - data-prep.py
4     - generate-content.R
5   post-render:
6     - cleanup.sh
7     - deploy.ts
```

Pre and Post Render Scripts

Pre and Post Render Scripts

Key Features

Pre and Post Render Scripts

Key Features

- Scripts run from **project root directory**.

Pre and Post Render Scripts

Key Features

- Scripts run from **project root directory**.
- **Re-computes metadata** after pre-render (can generate new files).

Pre and Post Render Scripts

Key Features

- Scripts run from **project root directory**.
- **Re-computes metadata** after pre-render (can generate new files).
- Supports **shell commands, Python, R, TypeScript, Lua**.

Pre and Post Render Scripts

Environment Variables Available

- `QUARTO_PROJECT_RENDER_ALL` - “1” if rendering entire project.
- `QUARTO_PROJECT_OUTPUT_DIR` - Output directory path.
- `QUARTO_PROJECT_INPUT_FILES` - List of input files (pre-render).
- `QUARTO_PROJECT_OUTPUT_FILES` - List of output files (post-render).

Pre and Post Render Scripts

Conditional Execution Example

● ● ●

PYTHON

```
1 # Only run expensive operations on full renders
2 import os
3 if os.environ.get("QUARTO_PROJECT_RENDER_ALL") == "1":
4     # Run expensive data processing
5     process_large_dataset()
```

Environment Variables

Environment Variables

Manage project environment variables with files and profiles:

Environment Variables

Manage project environment variables with files and profiles:

- **_environment** - Default environment variables for all renders.
- **_environment.local** - Local overrides (auto-ignored by Git).
- **_environment-{profile}** - Profile-specific variables.
- **_environment.required** - Documentation of required variables.

Variables

Variables

Insert dynamic content with shortcodes:

- `{{< var variable_name >}}` - From `_variables.yml` file.
- `{{< meta field_name >}}` - From document YAML metadata.
- `{{< env ENV_VARIABLE >}}` - From environment variables.

Variables

Insert dynamic content with shortcodes:

- `{{< var variable_name >}}` - From `_variables.yml` file.
- `{{< meta field_name >}}` - From document YAML metadata.
- `{{< env ENV_VARIABLE >}}` - From environment variables.

Markdown

MARKDOWN

```
1 - {{< meta title >}}
2 - {{< meta subtitle >}}
```

Variables

Insert dynamic content with shortcodes:

- `{{< var variable_name >}}` - From `_variables.yml` file.
- `{{< meta field_name >}}` - From document YAML metadata.
- `{{< env ENV_VARIABLE >}}` - From environment variables.

Markdown

MARKDOWN

1 - {{< meta title >}}

2 - {{< meta subtitle >}}

Output

- Formats & Projects
- Session 4

Project Profiles

Project Profiles

Adapt projects for different scenarios with configuration variants.

Project Profiles

Adapt projects for different scenarios with configuration variants.

What Are Profiles?

Project Profiles

Adapt projects for different scenarios with configuration variants.

What Are Profiles?

- Different configurations for the same project (production, development, basic/advanced versions).

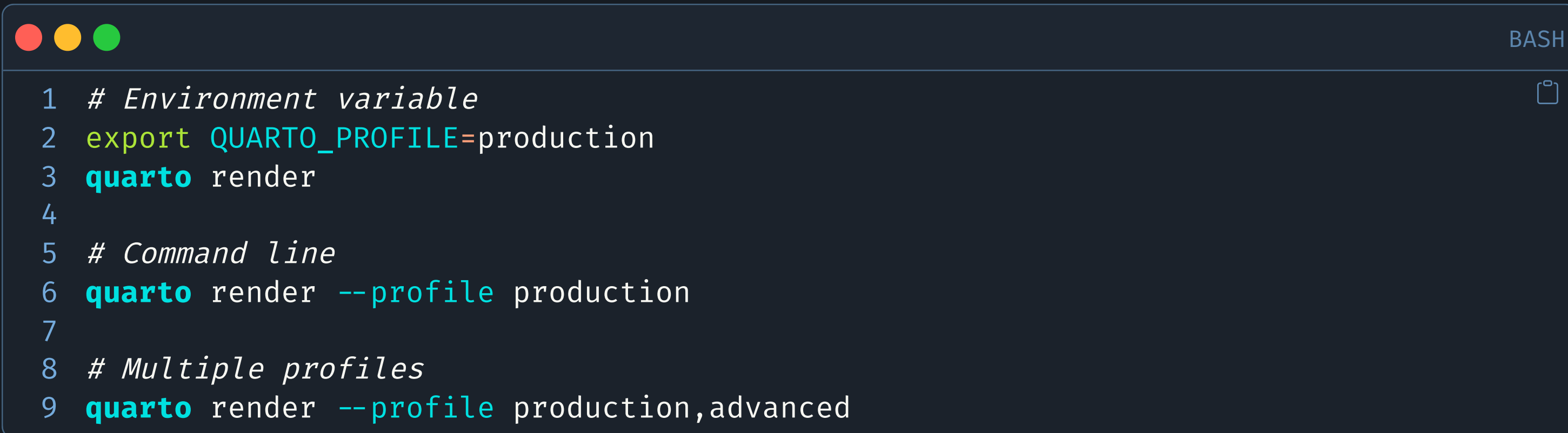
Project Profiles

Configuration Files

- `_quarto.yml` - Base configuration.
- `_quarto-{profile}.yml` - Profile-specific config that merges with base.
- `_environment-{profile}` - Profile-specific environment variables.

Project Profiles

Activation



```
1 # Environment variable
2 export QUARTO_PROFILE=production
3 quarto render
4
5 # Command line
6 quarto render --profile production
7
8 # Multiple profiles
9 quarto render --profile production,advanced
```

BASH



Project Profiles

Content Targeting



MARKDOWN

```
1 ::: {.content-visible when-profile="advanced"}
2 Advanced content only
3 :::
4
5 ::: {.content-hidden unless-profile="basic"}
6 Hidden for basic profile
7 :::
```



Project Profiles

Profile Groups & Defaults

YAML

```
1 profile:
2   group: [basic, advanced]
3   default: development
```

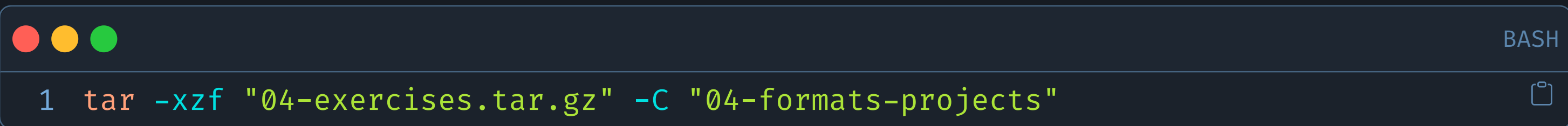
- Exercise Overview
- Part 1: Create a Project Structure
- Part 2: Multi-Format Configuration
- Part 3: Conditional Content
- Part 4: Test Multiple Formats
- Part 5: Add Project Features
- Success Criteria

Hands-On Exercise: Building Your First Project

Exercise Overview

Objective: Transform your computational portfolio from Session 3 into a structured project and explore multiple output formats.

Example Code:  [Exercises](#)



```
1 tar -xzf "04-exercises.tar.gz" -C "04-formats-projects"
```

BASH

Part 1: Create a Project Structure

Convert your portfolio into a project:

Website Project

Default Project

Book Project

Create a `_quarto.yml` file:

```
1 project:
2   type: website
3
4 website:
5   title: "My Learning Portfolio"
6   navbar:
7     left:
8       - href: index.qmd
9         text: Home
10      - about.qmd
11
12 format:
13   html:
14     theme: cosmo
15     toc: true
```

YAML



Part 1: Create a Project Structure

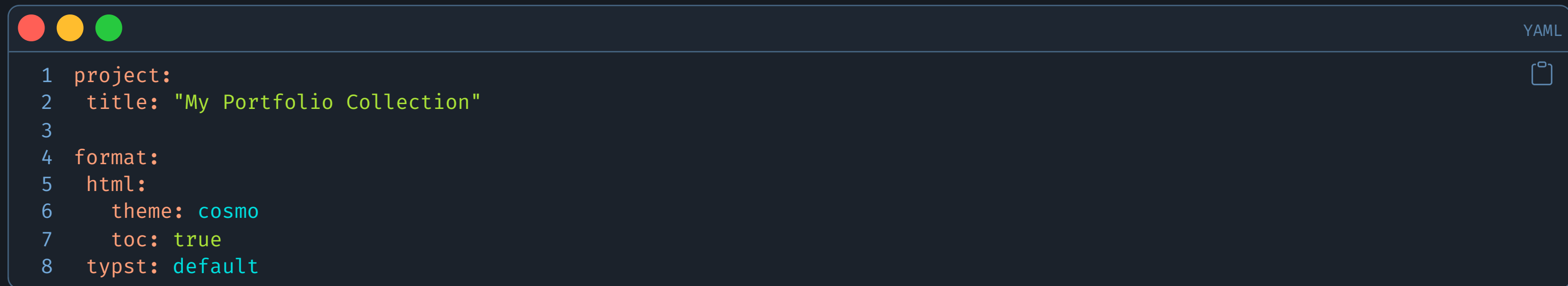
Convert your portfolio into a project:

Website Project

Default Project

Book Project

Create a `_quarto.yml` file:



```
1 project:
2   title: "My Portfolio Collection"
3
4 format:
5   html:
6     theme: cosmo
7     toc: true
8   typst: default
```

Part 1: Create a Project Structure

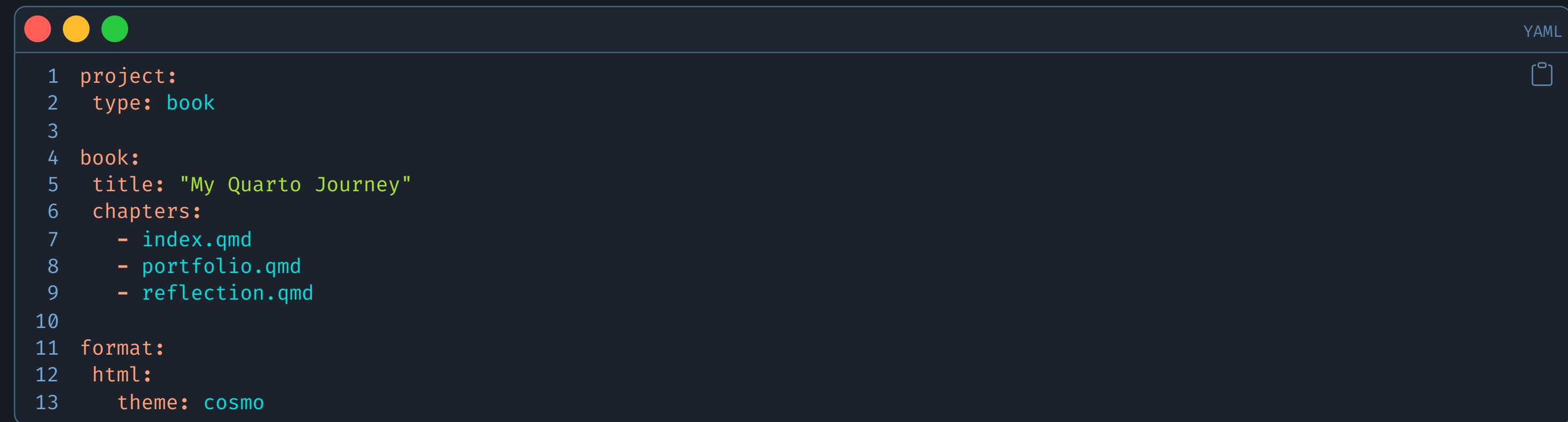
Convert your portfolio into a project:

Website Project

Default Project

Book Project

Create a `_quarto.yml` file:

A screenshot of a code editor window with a dark theme. The window has three colored window control buttons (red, yellow, green) in the top-left corner and a 'YAML' language identifier in the top-right corner. The code is written in a light blue/cyan color. It defines a project of type 'book' with a title 'My Quarto Journey' and three chapters: 'index.qmd', 'portfolio.qmd', and 'reflection.qmd'. It also sets the format to 'html' with the 'cosmo' theme.

```
1 project:
2   type: book
3
4 book:
5   title: "My Quarto Journey"
6   chapters:
7     - index.qmd
8     - portfolio.qmd
9     - reflection.qmd
10
11 format:
12   html:
13     theme: cosmo
```

Part 2: Multi-Format Configuration

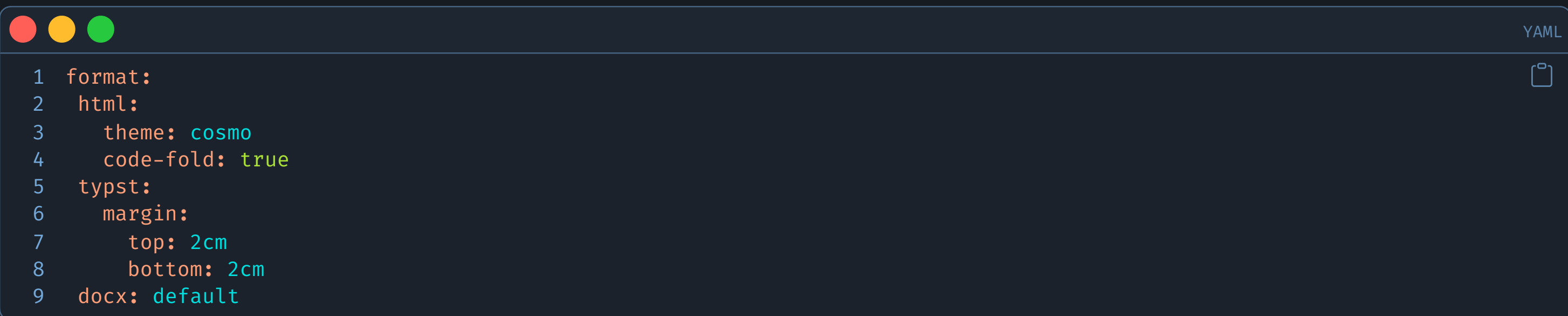
Add multiple output formats to your project:

Basic Multi-Format

Advanced Configuration

Presentation Format

Add to your `_quarto.yml`:



```
1 format:
2   html:
3     theme: cosmo
4     code-fold: true
5   typst:
6     margin:
7       top: 2cm
8       bottom: 2cm
9   docx: default
```

Part 2: Multi-Format Configuration

Add multiple output formats to your project:

Basic Multi-Format

Advanced Configuration

Presentation Format

Add format-specific options:

```
1 format:
2   html:
3     theme: cosmo
4     code-fold: true
5     fig-width: 8
6     fig-height: 6
7   typst:
8     fig-width: 6
9     fig-height: 4
10  pdf:
11    geometry:
12      - margin=1in
13    fontsize: 11pt
```

YAML



Part 2: Multi-Format Configuration

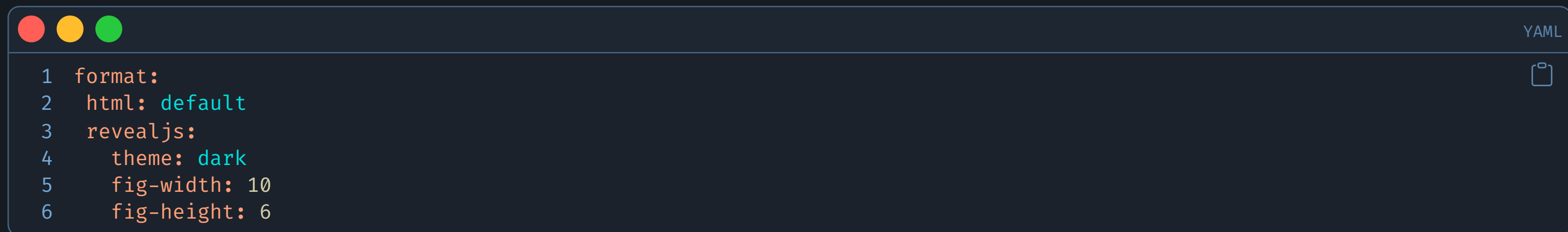
Add multiple output formats to your project:

Basic Multi-Format

Advanced Configuration

Presentation Format

Add slides to your outputs:



```
1 format:
2   html: default
3   revealjs:
4     theme: dark
5     fig-width: 10
6     fig-height: 6
```

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.

Part 3: Conditional Content

Add format-specific content to your documents:

Format-Specific Sections

Code Visibility by Format

Format-Specific Messages

Add to your content:

```
1 ::: {.content-visible when-format="html"}
2 This interactive content only appears in HTML output.
3 :::
4
5 ::: {.content-visible when-format="typst"}
6 This formatted text appears only in Typst/PDF output.
7 :::
8
9 ::: {.content-hidden when-format="revealjs"}
10 This detailed content is hidden in presentations.
11 :::
```

MARKDOWN



Part 3: Conditional Content

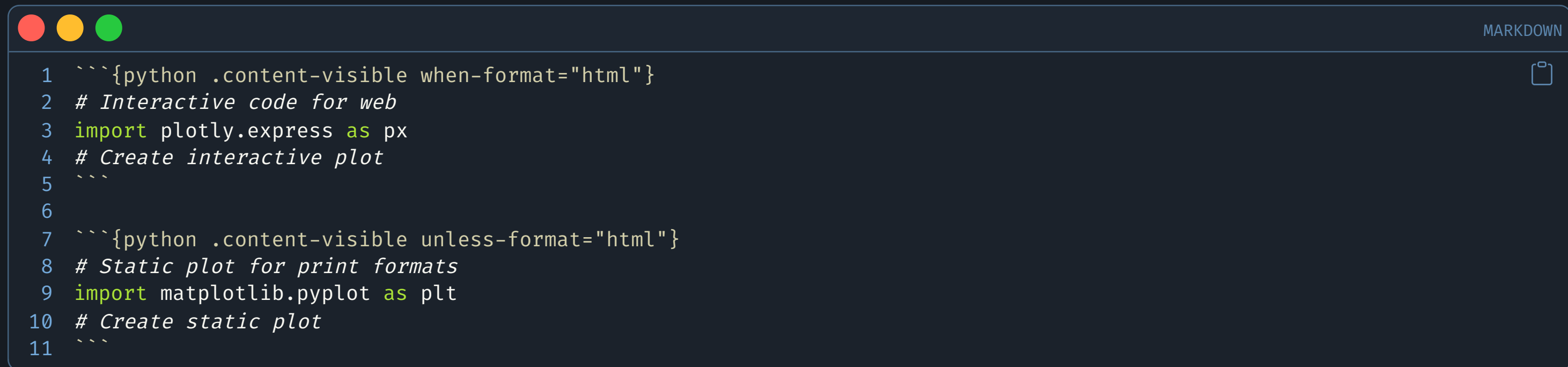
Add format-specific content to your documents:

Format-Specific Sections

Code Visibility by Format

Format-Specific Messages

Control code display:



```
1 ```{python .content-visible when-format="html"}
2 # Interactive code for web
3 import plotly.express as px
4 # Create interactive plot
5 ```
6
7 ```{python .content-visible unless-format="html"}
8 # Static plot for print formats
9 import matplotlib.pyplot as plt
10 # Create static plot
11 ```
```

Part 3: Conditional Content

Add format-specific content to your documents:

Format-Specific Sections

Code Visibility by Format

Format-Specific Messages

Add dynamic text:



MARKDOWN

```
1  [[View the interactive version online](link-to-html)]{.content-visible when-format="typst"}
2
3  [*This is the web version with interactive features*]{.content-visible when-format="html"}
```



Part 4: Test Multiple Formats

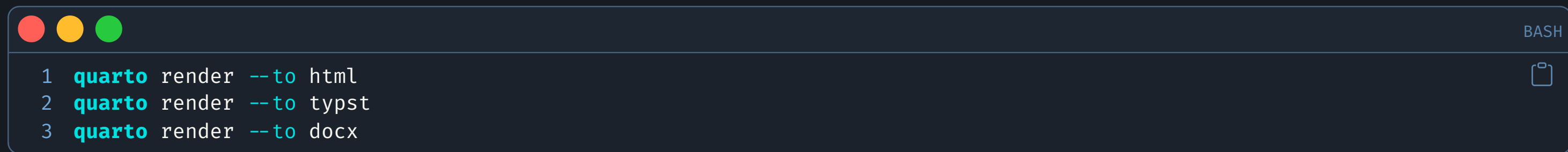
Render your project in different formats:

Single Format

All Formats

Preview Mode

Try rendering to specific formats:



```
1 quarto render --to html
2 quarto render --to typst
3 quarto render --to docx
```

The image shows a terminal window with a dark background. At the top left, there are three colored circles (red, yellow, green). At the top right, the word "BASH" is displayed. The terminal contains three lines of text, each starting with a line number (1, 2, 3) followed by the command "quarto render --to" and then the format "html", "typst", and "docx" respectively. A clipboard icon is visible on the right side of the terminal window.

Part 4: Test Multiple Formats

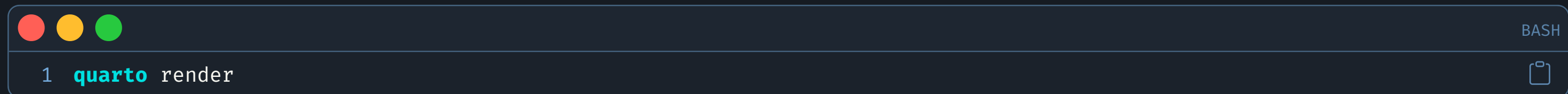
Render your project in different formats:

Single Format

All Formats

Preview Mode

Render all configured formats:



```
1 quarto render
```

A terminal window with a dark background. The title bar shows three colored circles (red, yellow, green) on the left and the text 'BASH' on the right. The main area contains the command '1 quarto render' with a line number '1' on the left. A clipboard icon is visible on the right side of the terminal area.

Part 4: Test Multiple Formats

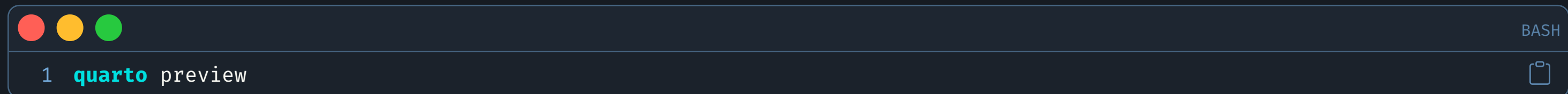
Render your project in different formats:

Single Format

All Formats

Preview Mode

Use preview for development:



```
1 quarto preview
```

A terminal window with a dark background. The title bar shows three colored circles (red, yellow, green) on the left and the text 'BASH' on the right. The main area shows a single line of text: '1 quarto preview'. On the right side of the terminal, there is a small icon of a document with a checkmark.

Part 5: Add Project Features

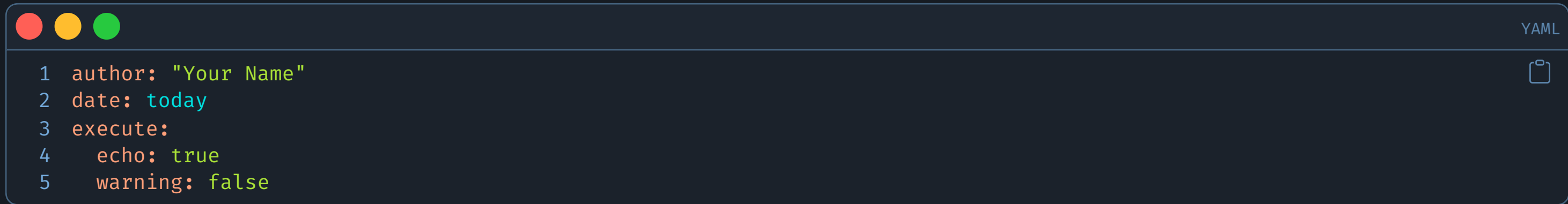
Enhance your project with additional features:

Shared Metadata

Resource Management

Environment Variables

Create `_metadata.yml`:



```
1 author: "Your Name"
2 date: today
3 execute:
4   echo: true
5   warning: false
```

Part 5: Add Project Features

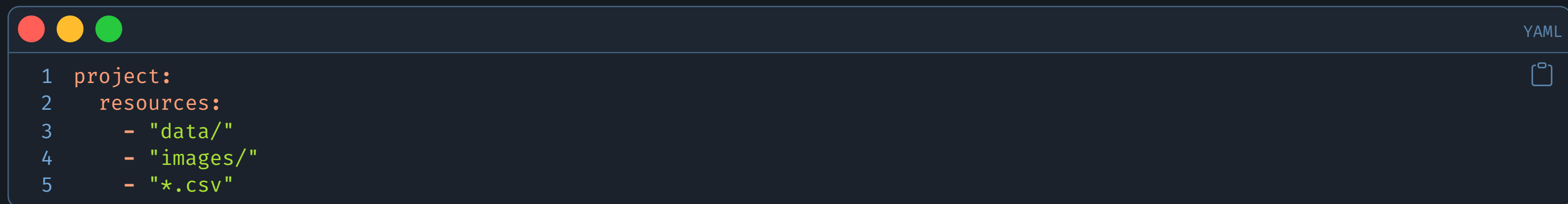
Enhance your project with additional features:

Shared Metadata

Resource Management

Environment Variables

Add to `_quarto.yml`:



```
1 project:
2   resources:
3     - "data/"
4     - "images/"
5     - "*.csv"
```

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.

Part 5: Add Project Features

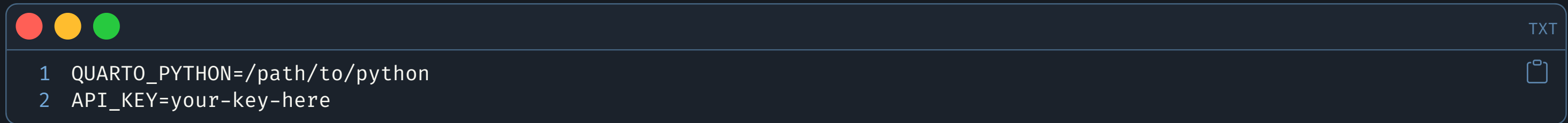
Enhance your project with additional features:

Shared Metadata

Resource Management

Environment Variables

Create `_environment` file:



```
1 QUARTO_PYTHON=/path/to/python
2 API_KEY=your-key-here
```

Success Criteria

You've successfully completed the exercise if you can:

- Create a properly structured project with `_quarto.yml`.
- Configure at least two different output formats.
- Add format-specific content that shows/hides appropriately.
- Successfully render your project to multiple formats.
- Test that conditional content works correctly.



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/in/MickaelCanouil)



Bluesky

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



Mastodon

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