

# Docco: Feature Showcase

Welcome to this demonstration of Docco's capabilities.

<a href="#">1 Key Features</a> .....	1
<a href="#">1.1 Frontmatter</a> .....	1
<a href="#">1.1.1 Supported Fields</a> .....	2
<a href="#">1.2 General Directive Rule</a> .....	3
<a href="#">1.3 Inlined Content</a> .....	3
<a href="#">1.3.1 Arguments</a> .....	3
<a href="#">1.3.2 Recursive Inlining</a> .....	3
<a href="#">1.4 Table of Contents</a> .....	3
<a href="#">1.5 Page Layout Directives</a> .....	4
<a href="#">1.5.1 Page Breaks</a> .....	5
<a href="#">1.5.2 Landscape Orientation</a> .....	6
<a href="#">1.5.3 Back to Portrait</a> .....	7
<a href="#">2 Headers &amp; Footers</a> .....	7
<a href="#">2.1 Configuration</a> .....	7
<a href="#">2.2 CSS Requirements</a> .....	7
<a href="#">2.3 Example Files</a> .....	8
<a href="#">3 Python Code Execution</a> .....	8
<a href="#">4 Conclusion</a> .....	9

## 1 Key Features

This document demonstrates Docco's core capabilities:

- Inline file inclusion with placeholder substitution
- Table of contents generation
- Page layout directives (page breaks and orientation changes)
- Headers and footers with directive support
- Multi-language support
- Python code execution

### 1.1 Frontmatter

YAML frontmatter at the beginning of the document (between --- delimiters) configures document processing:

```
---
```

**css**:

- "css/page.css"
- "css/toc.css"

**languages**: EN DE FR

```
--
```

### 1.1.1 Supported Fields

**css** - CSS stylesheet(s) for PDF styling. Can be:

- Single file (string): `css: "style.css"`
- Multiple files (inline array): `css: ["page.css", "theme.css"]`
- Multiple files (multiline list):

```
css:
```

- "css/page.css"
- "css/toc.css"

Paths are relative to the markdown file. Additional CSS can also be provided via the CLI `--css` argument, which overrides frontmatter styles.

**languages** - Space-separated language codes for multi-language document generation. When specified, Docco generates separate PDFs for each language, filtering content by `<!-- lang:CODE -->` tags.

Example:

```
languages: "EN DE"
```

See the [Multilingual Document Example](#) for a complete demonstration.

**header** and **footer** - HTML files for page headers and footers with placeholder and directive support:

```
header:  
  file: "header.html"  
  title: "My Document"  
  author: "John Doe"  
footer:  
  file: "footer.html"  
  title: "My Document"
```

The `file` key is required and specifies the HTML file path (relative to the markdown file). All other keys are placeholders that replace `{key}` in the HTML

file. Header/footer files support all directives (lang, inline, python), enabling dynamic content like dates and language-specific text. See the [Headers & Footers](#) section for details.

## 1.2 General Directive Rule

**Directives can appear anywhere in the document**, including in the middle of lines. However, directives inside code blocks (both inline `code` and fenced blocks) are **protected** and will not be processed. This allows you to show directive syntax as examples in documentation without triggering them.

## 1.3 Inlined Content

The **inline** directive embeds external markdown files with placeholder substitution.

Syntax: <!-- inline:"foobar.md" key1="value1" key2="value2" -->

### 1.3.1 Arguments

All attributes after the path are placeholders. For example, `author="Docco Team"` replaces `{ {author} }` in the inlined file. Arguments are optional. Spaces around the `inline` keyword and colon are accepted. See example below:

This content is inlined, with arguments.

Author: Docco Team Date: 2025-10-26

### 1.3.2 Recursive Inlining

Inlined markdown files can themselves contain inline directives, allowing for multi-level composition (up to a maximum depth of 10 levels to prevent infinite recursion). This enables modular document structures where content can be composed from multiple nested files.

## 1.4 Table of Contents

The `<!-- TOC -->` directive generates a hierarchical table of contents with automatic numbering (1, 1.1, 1.2, etc.).

Use `<!-- toc:exclude -->` before a heading to exclude it from the TOC and remove its numbering.

## 1.5 Page Layout Directives

## 1.5.1 Page Breaks

This section starts on a **new page** using the `<! -- page break -->` directive.

## 1.5.2 Landscape Orientation

This section uses **landscape orientation** with the `<!-- landscape -->` directive, providing more horizontal space for wide content.

### 1.5.3 Back to Portrait

This section returns to **portrait orientation** using the `<!-- portrait -->` directive.

## 2 Headers & Footers

Docco supports page headers and footers using HTML files with full directive support. Headers and footers are processed through the same pipeline as the main document, allowing for:

- Placeholder substitution
- Language-specific content with `<!-- lang:CODE -->` directives
- Dynamic content with `<!-- python -->` directives
- File inclusion with `<!-- inline -->` directives

### 2.1 Configuration

Headers and footers are configured in the frontmatter:

```
header:  
  file: "header.html"  
  title: "Docco Feature Showcase"  
  author: "Docco Team"  
footer:  
  file: "footer.html"  
  title: "Docco"
```

The `file` key specifies the HTML file path. All other keys are placeholders that replace `{{key}}` in the HTML file.

### 2.2 CSS Requirements

Headers and footers require CSS to position them on the page using CSS Paged Media. Example:

```
@page {  
  margin-top: 2.5cm;  
  margin-bottom: 2.5cm;  
  @top-center {  
    content: element(header);
```

```
    }
    @bottom-center {
        content: element(footer);
    }
}
```

See examples/css/header\_footer.css for a complete example.

## 2.3 Example Files

**Header (header\_showcase.html):**

```
<div style="text-align: center; color: #666;">
    {{title}} - {{author}}
</div>
```

**Footer (footer\_showcase.html):**

```
<div style="display: flex; justify-content: space-between;">
    <span>{{title}} - Generated: <!-- python -->
import datetime
print(datetime.datetime.now().strftime("%Y-%m-%d"))
    <!-- /python --></span>
    <span class="page-number"></span>
</div>
```

For language-aware headers and footers with `<!-- lang:CODE -->` directives, see the [Multilingual Document Example](#).

## 3 Python Code Execution

The `<!-- python -->` directive executes Python code and inserts the stdout output into the markdown. This is useful for generating dynamic content.

The output can contain other directives (markdown, inline files, etc.).

**Important:** Python code execution is disabled by default for security reasons. Use the `--allow-python` flag to enable it.

Syntax: `<!-- python -->code<!-- /python -->`

**Example - a loop that outputs digits**

This code:

```
print("_", end=' ')
for i in range(10):
    print(i, end=' ')
print("_", end='')
```

Returns: 0123456789

## 4 Conclusion

This document demonstrates all Docco's capabilities.