Markdown Styles Demo

This document demonstrates all the available Bootstrap-inspired styles for educational content. Use this as a reference when creating your course materials.

Table of Contents

Page Break Example:

The horizontal rule above uses a custom HTML `<div style="page-break-after: always;"></div>` to force a page break in print or PDF output.

You can insert this anywhere in your Markdown to separate sections onto new pages.

Paragraphs and Line Break Examples

You can create line breaks in Markdown using two spaces at the end of a line, or by using the
 HTML tag.

Example using two spaces:
This is the first line.
This is the second line.

**Example using
 tag:**
This is the first line.
This is the second line.

Within lists or tables:

- Item one
  continues on the next line
- Item two
  with a manual break

Inside block quotes:

> This is a blockquote line one.
> This is line two.

Different methods of inserting newlines:

Newlines within a paragraph in HTML are created using
.
In an .md file, the markdown standard solution is to use at the end of the line.
In VS Code, you can use the \ character at the end of a line to force a line break instead – While this technically works in VS Code, standard Markdown processors may not support it, so it's less portable than using two trailing spaces or
.

Tip

Use two spaces at the end of a line for compatibility with most Markdown processors.

Author's preference:

Mr. Eli (the author) prefers using \ for line breaks instead of trailing spaces for clarity and ease of editing in VS Code.

C++ Syntax
Perhaps the main reasoning for this is that in C++, the newline character used is the standard output character for **bash** \n or **powershell** \r.
The \ character resembles this format.

Educational Content Blocks

Below are examples of each Obsidian callout type, rendered with Bootstrap-inspired styles using the Markdown Obsidian Callout extension.

Warning Blocks

> **Warning**
>
> With the Markdown Obsidian Callout extension, warning blocks are rendered with Bootstrap-inspired warning styles (orange theme), making alerts visually prominent and easy to spot.

Tip Blocks

> **Tip**
>
> Tip blocks are enhanced with Bootstrap info (cyan theme) styling, providing helpful advice with clear visual separation, thanks to the Obsidian Callout extension.

Info Blocks

Note: *This block is expandable*

> **Info** >

Danger/Error Blocks

> **Danger**
>
> Danger blocks receive a stronger red Bootstrap theme, emphasizing critical warnings and urgent messages with distinct styling from the callout extension.

> **Error**
>
> Error blocks highlight problems or failures with a bold red theme, making issues easy to identify.

Success Blocks

> **Success**
>
> Success blocks are styled with Bootstrap success (green theme) colors, offering positive reinforcement and confirmation, visually highlighted by the Obsidian Callout extension.

Check

This is a check callout, confirming a completed action or success.

Bug Blocks

Bug

Bug callouts use a distinct style to highlight known issues or defects in code or documentation.

Default/Note Blocks

Note

Default note blocks provide general information or reminders, styled with a neutral Bootstrap theme.

Example Blocks

Example

Example callouts showcase sample code, usage, or scenarios for better understanding.

Fail/Missing Blocks

Fail

Fail blocks indicate missing features, failed tests, or incomplete sections.

Missing

This callout marks content that is not yet available or needs to be added.

Important Blocks

Important

Important callouts emphasize critical instructions or must-read information.

Question/Help/FAQ Blocks

Question

Question callouts highlight frequently asked questions or prompts for further discussion.

Help

Help callouts provide guidance or troubleshooting tips.

Summary/Abstract/TLDR Blocks

Summary

Summary callouts give a brief overview or abstract of the section.

Tldr

TLDR (Too Long; Didn't Read) callouts summarize key points for quick reference.

Todo Blocks

Todo

Todo callouts list tasks or items that need attention or completion.

Quote/Cite Blocks

Quote

Quote callouts are used for citations, references, or notable statements.

Cite

Cite callouts reference sources or important literature.

Fenced Spans

What Are Fenced Spans?

Fenced spans are an extension to standard Markdown that allow you to apply custom classes or attributes to inline or block-level content. This is especially useful for advanced styling, such as highlighting, warnings, or other semantic cues, directly within your Markdown.

Syntax Example

You can use the following syntax to create a fenced span:

```
[fenced-span]{.fenced}
```

> renders as:
>
> fenced-span

This will render the text `fenced-span` with the CSS class `fenced`, allowing you to style it via your custom CSS.

Use Cases

- **Highlighting important terms**:
  `[important]{.highlight}`
- **Semantic labeling**:
  `[deprecated]{.warning}`
- **Custom inline styling**:
  `[inline code]{.code-style}`

Requirements

To use fenced spans, your Markdown processor must support the Markdown-it Attribute extension or a similar plugin. In VS Code, this is typically enabled by the Markdown Extended extension or compatible plugins.

> **Tip**
>
> Fenced spans are a powerful way to add semantic meaning and custom styling to your Markdown content without resorting to raw HTML.

Typography Demonstrations

Heading 1 (2.5rem)

Heading 2 (2rem)

Heading 3 (1.75rem)

Heading 4 (1.5rem)

Heading 5 (1.25rem)

Heading 6 (1rem)

Text Formatting

This is **bold text** and this is *italic text*. You can also use highlighted text for emphasis. This is underlined text using HTML <u> tags or _ characters. This is ~~strikethrough text~~ using double tildes ~~`text`~~.

Inline Code Styling

Here are examples of styled inline code:

`code.warning` - Warning-styled
`code.tip` - Tip-styled
`code.info` - Info-styled
`code.error` - Error-styled
`code.success` - Success-styled

> **Warning**
>
> Be careful of longer line code: `Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer euismod quam consectetur velit sodales, vel euismod ipsum volutpat. Phasellus et lacus ligula. Duis mattis semper ligula et egestas. Etiam molestie eget nulla nec sodales. Praesent euismod ex in purus ornare dictum. Etiam fringilla malesuada neque, ac dignissim nibh laoreet nec. Vestibulum sapien mauris, pharetra vel tortor nec, pharetra euismod ipsum. Curabitur vitae accumsan sem, ut suscipit orci. Etiam sed nunc dignissim, aliquet quam non, dictum felis.` `Aenean quam ex, ullamcorper ut imperdiet quis, egestas ut quam. Phasellus fringilla efficitur sem eu blandit. Etiam feugiat nisi ex, nec maximus leo molestie id. Nullam gravida neque eros, imperdiet rutrum enim maximus a. Duis ullamcorper ante eget risus congue, et interdum augue luctus. Ut fermentum massa et eros sollicitudin commodo. Integer condimentum pellentesque tortor, eget egestas magna tristique eu. Fusce neque eros, efficitur a sem non, porttitor imperdiet lectus. Quisque pharetra odio et ante vestibulum ornare. Mauris tempus metus id enim tincidunt, et ornare ipsum fermentum. Nam laoreet placerat erat quis`

rhoncus. Donec vitae tempor nisl, sed fringilla purus. Vestibulum sed ligula enim. Morbi pharetra mattis porta.

Code Blocks

We've styled the code blocks marked with classnames *(which allows for deeper styling)*. All code blocks below are styled to match their corresponding callout types, demonstrating how custom CSS classes can visually differentiate code for warnings, tips, errors, and more.[1]

Warning

```
#include <iostream>
#include <string>

class Student {
private:
    std::string name;
    int id;
    // ...
}
```

Block quotes

> Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer euismod quam consectetur velit sodales, vel euismod ipsum volutpat. Phasellus et lacus ligula. Duis mattis semper ligula et egestas. Etiam molestie eget nulla nec sodales. Praesent euismod ex in purus ornare dictum. Etiam fringilla malesuada neque, ac dignissim nibh laoreet nec. Vestibulum sapien mauris, pharetra vel tortor nec, pharetra euismod ipsum. Curabitur vitae accumsan sem, ut suscipit orci. Etiam sed nunc dignissim, aliquet quam non, dictum felis. Aenean quam ex, ullamcorper ut imperdiet quis, egestas ut quam. Phasellus fringilla efficitur sem eu blandit. Etiam feugiat nisi ex, nec maximus leo molestie id.
>
> Nullam gravida neque eros, imperdiet rutrum enim maximus a. Duis ullamcorper ante eget risus congue, et interdum augue luctus. Ut fermentum massa et eros sollicitudin commodo. Integer condimentum pellentesque tortor, eget egestas magna tristique eu. Fusce neque eros, efficitur a sem non, porttitor imperdiet lectus. Quisque pharetra odio et ante vestibulum ornare. Mauris tempus metus id enim tincidunt, et ornare ipsum fermentum. Nam laoreet placerat erat quis rhoncus. Donec vitae tempor nisl, sed fringilla purus. Vestibulum sed ligula enim. Morbi pharetra mattis porta.

Block Quotes alternative using the obsidian callout extension:

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer euismod quam consectetur velit sodales, vel euismod ipsum volutpat. Phasellus et lacus ligula. Duis mattis semper ligula et egestas. Etiam molestie eget nulla nec sodales. Praesent euismod ex in purus ornare dictum. Etiam fringilla malesuada neque, ac dignissim nibh laoreet nec. Vestibulum sapien mauris, pharetra vel tortor nec, pharetra euismod ipsum. Curabitur vitae accumsan sem, ut suscipit orci. Etiam sed nunc dignissim, aliquet quam non, dictum felis. Aenean quam ex, ullamcorper ut imperdiet quis, egestas ut quam. Phasellus fringilla efficitur sem eu blandit. Etiam feugiat nisi ex, nec maximus leo molestie id.

Nullam gravida neque eros, imperdiet rutrum enim maximus a. Duis ullamcorper ante eget risus congue, et interdum augue luctus. Ut fermentum massa et eros sollicitudin commodo. Integer condimentum pellentesque tortor, eget egestas magna tristique eu. Fusce neque eros, efficitur a sem non, porttitor imperdiet lectus. Quisque pharetra odio et ante vestibulum ornare. Mauris tempus metus id enim tincidunt, et ornare ipsum fermentum. Nam laoreet placerat erat quis rhoncus. Donec vitae tempor nisl, sed fringilla purus. Vestibulum sed ligula enim. Morbi pharetra mattis porta.

Tables with Bootstrap Styling

Feature	Description	Status
Warning Blocks	Red-themed alerts	✅ Active
Tip Blocks	Cyan-themed helpful hints	✅ Active
Info Blocks	Blue-themed information	✅ Active
Success Blocks	Green-themed confirmations	✅ Active
Enhanced Tables	Bootstrap-styled tables	✅ Active

Complex Table Example

Unit	Chapter	Topic	Difficulty	Estimated Time
1	1.0	Using Objects	Beginner	2 hours
1	1.1	Defensive Programming	Intermediate	3 hours
2	2.0	Encapsulation Design	Intermediate	4 hours
2	2.1	Building a Class	Advanced	5 hours

Not yet supported: Grid Tables

```
% pandoc -f rst
+------------------------+------------+----------+----------+
| Header row, column 1   | Header 2   | Header 3 | Header 4 |
| (header rows optional) |            |          |          |
+========================+============+==========+==========+
| body row 1, column 1   | column 2   | column 3 | column 4 |
+------------------------+------------+----------+----------+
| body row 2             | Cells may span columns.          |
+------------------------+------------+--------------------+
| body row 3             | Cells may  | - Table cells      |
+------------------------+ span rows. | - contain          |
| body row 4             |            | - body elements.   |
+------------------------+------------+--------------------+
%
```

Support for pandoc-style grid tables not available yet. Maybe there's an extension that I missed.

1	2	3	4	5	6	7	8
9	10	11	12	13	14	14	13
15	16	17	18	19	20	21	22

23	24	25	26	27	28

Code Examples

C++ Code Block with Syntax Highlighting

```cpp
// Green block class added using ```cpp{.success-style}
#include <iostream>
#include <string>

class Student {
private:
    std::string name;
    int id;

public:
    // Constructor
    Student(const std::string& studentName, int studentId)
        : name(studentName), id(studentId) {}

      // Accessor methods
    std::string getName() const { return name; }
    int getId() const { return id; }

      // Display method
    void display() const {
        std::cout << "Student: " << name
            << " (ID: " << id << ")"
            << std::endl;
    }
};

int main() {
    Student student("John Doe", 12345);
    student.display();
    return 0;
}
```

Python Code Example

```python
class Calculator:
    def __init__(self):
        self.result = 0

      def add(self, value):
        self.result += value
        return self

      def multiply(self, value):
        self.result *= value
        return self

      def get_result(self):
        return self.result

# Usage example
calc = Calculator()
result = calc.add(5).multiply(3).get_result()
print(f"Result: {result}")  # Output: Result: 15
```

JSON Configuration Example

```json
{
  "markdown.styles": [
    "./.vscode/markdown-styles.css",
  ],
  "markdown.preview.breaks": true,
  "markdown.preview.linkify": true,
  "markdown.preview.typographer": true
}
```

Mathematical Expressions

Inline Math

The quadratic formula is $x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$.

Block Math Equations

$$f(x) = ax^2 + bx + c \tag{1}$$

$$f'(x) = 2ax + b \tag{2}$$

$$\int f(x)\,dx = \frac{ax^3}{3} + \frac{bx^2}{2} + cx + C \tag{3}$$

Algorithm Complexity

$$\begin{cases} O(1) & \text{constant time} \\ O(\log n) & \text{logarithmic time} \\ O(n) & \text{linear time} \\ O(n \log n) & \text{linearithmic time} \\ O(n^2) & \text{quadratic time} \end{cases}$$

Interactive Elements

Collapsible Sections

▶ Click to expand: Advanced C++ Topics
▶ Click to expand: Design Patterns

Definition lists

HTML Example (Definition List)

You can embed raw HTML directly in Markdown to create advanced definition lists.

```
<dl>
    <dt>Apple</dt>
    <dd>
        Pomaceous fruit of plants of the genus Malus in
        the family Rosaceae.
    </dd>
    <dd>An American computer company.</dd>
    <dt>Orange</dt>
    <dd>The fruit of an evergreen tree of the genus Citrus.</dd>
</dl>
```

There is a shorthand for this using markdown:

```
Apple
:   Pomaceous fruit of plants of the genus Malus in
    the family Rosaceae. <!--can be multiple lines-->
:   An American computer company.

Orange
:   The fruit of an evergreen tree of the genus Citrus.
```

Rendered Output:

Apple
: Pomaceous fruit of plants of the genus Malus in the family Rosaceae.
: An American computer company.

Orange

The fruit of an evergreen tree of the genus Citrus.

Educational Character Sections

Sue's Tips

Sue's Tips

Always use meaningful variable names and consistent formatting. Your future self (and your teammates) will thank you. Focus on writing code that works first, then optimize for readability.

Sam's Corner

Sam's Corner

Did you know that C++ templates are Turing complete? This means you can theoretically compute any computable function at compile time using only template metaprogramming techniques.

Lists and Organization

Ordered Lists

1. **First Principle**: Encapsulation
    - Hide implementation details
    - Provide clean interfaces
    - Protect data integrity
2. **Second Principle**: Inheritance
    - Code reusability
    - Hierarchical relationships
    - Polymorphic behavior
3. **Third Principle**: Polymorphism
    - Runtime type binding
    - Interface consistency

- Flexible design patterns

Unordered Lists

- ✅ **Completed Topics**

    - Basic syntax and semantics
    - Control structures and functions
    - Arrays and pointers
- 🔄 **In Progress**
    - Object-oriented design
    - Template programming
    - STL containers
- 📋 **Upcoming**
    - Advanced templates
    - Design patterns
    - Performance optimization

Todo List (unordered list with brackets)

- [x] ✅ **Completed Topics**
    - [x] Basic syntax and semantics
    - [x] Control structures and functions
    - [ ] Arrays and pointers
- [ ] 🔄 **In Progress**
    - [ ] Object-oriented design
    - [ ] Template programming
    - [ ] STL containers
- [ ] 📋 **Upcoming**
    - [ ] Advanced templates
    - [ ] Design patterns
    - [ ] Performance optimization

Task Lists

- [x] Set up development environment
- [x] Learn basic C++ syntax
- [ ] Master class design
- [ ] Implement inheritance hierarchies
- [ ] Practice template programming
- [ ] Complete final project

Conclusion

This demo showcases all available Bootstrap-enhanced Markdown styles. The styles provide:

Benefits:

Professional appearance, consistent formatting, improved readability, and enhanced educational content presentation.

Quick Reference

Style Type	Syntax	Use Case
Warning	`> [!WARNING]` (quote block)	Important alerts
	`` `WARNING`{.warning} `` (code block)	
Tip	`> [!TIP]` or `` `TIP`{.tip} ``	Helpful advice
Info	`> [!INFO]` or `` `INFO`{.info} ``	Additional context
Fail	`> [!Fail]` or `` `Fail`{.fail} ``	Critical warnings
Success	`> [!SUCCESS]` or `` `SUCCESS`{.success} ``	Positive feedback
Highlight	`==text==`	Emphasis
Math	`$equation$` or `$$block$$`	$Mathematical\ Expressions$

Happy writing with your enhanced Markdown styles! 🎉

```
void helloWorld() {
    return;
}
```

Warning

Your warning message here

Exporting to PDF

This document demonstrates a complete “What You See Is What You Get” (WYSIWYG) PDF export workflow using VS Code with the Markdown Extended extension. The workflow combines dynamic CSS styling, frontmatter metadata, and custom footer templates for professional document generation.

🔧 Workflow Components

1. Markdown Frontmatter

Each document includes structured metadata in YAML format:

```
---
title: "Bootstrap-Enhanced Markdown Styles Demo"
course: "CS 165"
unit: "Demo"
chapter: "Styles-Demo"
tags: [markdown, bootstrap, styles]
---
```

2. Dynamic CSS Variables

The `markdown-styles.css` file uses CSS custom properties for dynamic footer content:

Bug

```
:root {
    --course-name: "CS 165";
    --unit-name: "Demo Unit";
    --chapter-name: "Styles Demo";
}

@page {
    @bottom-center {
        content: "Page " counter(page) " | " var(--chapter-name) " | " var(--
        font-size: 9px;
    }
}
```

Todo

Create a system that dynamically pulls the md frontmatter into our `:root` selector in css

How does this work? >

3. VS Code Settings Configuration

The `.vscode/settings.json` includes:

```
{
    "markdown.styles": ["./markdown-styles.css"],
    "markdownExtended.pdfFooterTemplate": "<!-- Optional fallback -->"
}
```

4. Export Process

1. **Open** your markdown file in VS Code
2. **Ensure** the workspace includes the `.vscode` folder with `markdown-styles.css`

> **Note**
>
> You will have to individually open up every folder containing the `Note.vscode` folder for this to work

3. **Use Command Palette** (`Ctrl+Shift+P`) → "Markdown Extended: Export (pdf)"
4. **Result**: Professional PDF with custom footers, styling, and formatting

✨ Key Benefits

- **WYSIWYG**: Preview exactly matches PDF output
- **Dynamic Footers**: Page numbers, chapter names, unit info automatically included
- **Professional Styling**: Bootstrap-inspired callouts, tables, and typography
- **Consistent Branding**: Unified appearance across all course documents
- **Frontmatter Integration**: Document metadata drives styling decisions

🔄 Future Enhancements

- **Automated Script**: Parse frontmatter to auto-generate CSS variables
- **Template System**: One CSS file with dynamic frontmatter extraction
- **Batch Processing**: Export multiple documents with consistent styling

> **Pro Tip**
>
> The CSS `@page` rules take precedence over `markdownExtended.pdfFooterTemplate`, so you can use pure CSS for maximum control over footer formatting.

All code blocks styled like the callout types

```bug
bug
```

```default
default
```

```error
error
```

```example
example
```

Fail

fail

Important

important

Info

info

Question

question

Success

success

Summary

summary

Tip

tip

Todo

todo

Warning

warning

Quote

quote

```
bug-style
```

```
default-style
```

```
error-style
```

```
example-style
```

```
fail-style
```

```
important-style
```

```
info-style
```

```
question-style
```

```
success-style
```

```
summary-style
```

```
tip-style
```

```
todo-style
```

```
warning-style
```

```
quote-style
```

Case Sensitivity >

This is an bug callout.

This is an default callout.

This is an error callout.

This is an example callout.

This is an fail callout.

This is an important callout.

This is an info callout.

This is an question callout.

This is an success callout.

This is an summary callout.

This is an tip callout.

This is an todo callout.

This is an warning callout.

This is an quote callout.

Callouts are useful for providing additional context.

You can use callouts to highlight important information.

Callouts support Markdown formatting, **bold**, *italic*, and `inline code`.

Eallouts can include lists:

- Point one
- Point two
- Point three

> callouts can contain links: Learn more

> This is an expandable info callout.

> callouts can be nested:
>
> > Nested callout example.

> You can use emojis in callouts 🚀

> callouts help organize content and improve readability.

Complete Style Reference & Requirements

This section documents all the crucial criteria that must be propagated to other `.md` files in this workspace and reflected in the `.code-snippets` file.

📋 Mr. Eli's Preferences & Standards

Line Breaks

> **Author's Preference:**
>
> Mr. Eli (the author) prefers using \ for line breaks instead of trailing spaces for clarity and ease of editing in VS Code.\\ **Reasoning:** In C++, the newline character used is the standard output character for **bash** \n or **powershell** `\r`. The \ character resembles this format.

Examples:

- Primary method: `Line one\`
- Alternative: `Line one` (two spaces)
- HTML method: `Line one<br>`

Required VS Code Configuration

```
{
    "markdown.styles": [
        "./.vscode/markdown-styles.css"
    ],
    "markdown.preview.breaks": false,
    "markdown.preview.linkify": true,
    "markdown.preview.typographer": true
}
```

> ⚠ **No Bootstrap or Git Dependencies**
>
> All styles must be accessible to the IDE with only the above configuration. No external dependencies allowed.

🎯 Complete Feature Checklist

Educational Blocks/Callouts (All 18 Types)

- [x] `[!WARNING]` - Orange theme alerts
- [x] `[!TIP]` - Cyan theme helpful hints
- [x] `[!INFO]` - Blue theme information
- [x] `[!DANGER]` - Red theme critical warnings
- [x] `[!ERROR]` - Red theme error messages
- [x] `[!SUCCESS]` - Green theme confirmations
- [x] `[!CHECK]` - Green theme completed actions
- [x] `[!BUG]` - Distinct style for known issues
- [x] `[!NOTE]` - Neutral theme general information
- [x] `[!EXAMPLE]` - Sample code/usage scenarios
- [x] `[!FAIL]` - Missing features/failed tests
- [x] `[!MISSING]` - Content not yet available
- [x] `[!IMPORTANT]` - Critical instructions
- [x] `[!QUESTION]` - FAQ or discussion prompts
- [x] `[!HELP]` - Guidance/troubleshooting
- [x] `[!SUMMARY]` - Brief overview/abstract
- [x] `[!TLDR]` - Quick reference summaries
- [x] `[!TODO]` - Tasks needing attention
- [x] `[!QUOTE]` - Citations/references
- [x] `[!CITE]` - Source references

Inline Code Styling

```
`code.warning`{.warning} - Warning-styled code\
`code.tip`{.tip} - Tip-styled code\
`code.info`{.info} - Info-styled code\
`code.error`{.error} - Error-styled code\
`code.success`{.success} - Success-styled code\
`code.fail`{.fail} - Fail-styled code\
`code.bug`{.bug} - Bug-styled code\
`code.example`{.example} - Example-styled code\
`code.important`{.important} - Important-styled code\
`code.default`{.default} - Default-styled code
```

Block Quotes (Standard vs Callout Extension)

Standard Block Quote:

```
> Lorem ipsum dolor sit amet, consectetur adipiscing elit.\
>\
> Second paragraph with backslash line breaks.
```

Callout Extension Alternative:

```
> [!INFO] Block Quotes alternative using the obsidian callout extension:
> Lorem ipsum with enhanced styling and visual separation.\
>\
> Second paragraph maintains callout theme formatting.
```

Mathematical Expressions

- **Inline Math:** `$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$`
- **Block Math:** `$$equation$$`
- **Aligned Equations:** `\begin{align}...\end{align}`
- **Cases:** `\begin{cases}...\end{cases}`

Educational Character Sections

Sue's Tips:

```
> [!DANGER] 💡 **Sue's Tip:**
> ![Sue](.vscode/assets/sue.png){width=50 align=right}
> Practical advice for getting the job done efficiently
```

Sam's Corner:

```
> [!TLDR] 🤓 **Sam's Corner:**
> ![Sam](.vscode/assets/sam.png){width=50 align=right}
> Technical details and interesting tidbits
```

Task Lists with Brackets

```
> [!TODO] Todo List (unordered list with brackets)
> - [x] ✅ **Completed Topics**
>   - [x] Basic syntax and semantics
>   - [x] Control structures and functions
>   - [ ] Arrays and pointers
>
> - [ ] 🔄 **In Progress**
>   - [ ] Object-oriented design
>   - [ ] Template programming
```

Code Blocks Styled Like Callout Types

All code blocks can be styled to match callout types:

````
```{.warning}
warning-styled code block


tip-style code block
```
````

> Info
>
> info-styled code block

````
#### **Complete Footnote System**
- **Reference:** `[^1]`
- **Definition:** `[^1]: Footnote content here`

#### **Frontmatter Template**
```yaml
---
title: "Document Title"
description: "Brief description"
course: "CS 165"
unit: "Unit X"
chapter: "Chapter Name"
tags:
  - cpp
  - object-oriented-programming
source:
  type: "AI-Generated Draft"
  method: "Manual Creation"
  original: "source.pdf"
  generated: "2025-08-04 16:30:00"
author: "CS 165 Course Materials"
date: "2025-08-04"
---
````
````

Table of Contents Structure

```
| Table of Contents |
| :--- |
| Course Overview |
| [Section Name](path/to/file.md) |
| [Another Section](path/to/another.md) |
```

🔧 Implementation Requirements

1. **All `.md` files** must include proper frontmatter
2. **All callout types** must be consistently used
3. **Line breaks** must use \ (backslash) method
4. **Code snippets** must be available for all features
5. **CSS styling** must work with specified VS Code settings only
6. **PDF export** must maintain all styling and formatting
7. **Character sections** (Sue/Sam) must include proper image references
8. **Mathematical expressions** must render properly in preview and PDF

📚 Code Snippets Coverage

The `.vscode/markdown.code-snippets` file includes shortcuts for:

- All 18 callout types (`warning`, `tip`, `info`, etc.)
- Line break methods (`linebreak`, `linebreak-spaces`, `linebreak-html`)
- Text formatting (`bold`, `italic`, `strike`, `highlight`)
- Code blocks (`codeblock`, `codestyle`, `allstyles`)
- Tables (`table`, `table-aligned`, `htmltable`)
- Math (`math`, `imath`, `mathalign`, `mathcases`)
- Educational content (`eduheader`, `sues-tips`, `sams-corner`)
- Document structure (`toc`, `pagebreak`, `hr`)
- Interactive elements (`collapse`, `deflist`, `tasklist`)

> ✓ **Complete Implementation**
>
> This comprehensive reference ensures consistent, professional educational content across the entire workspace with Mr. Eli's preferred formatting standards.

---

1. All code blocks below are styled to match their corresponding callout types, demonstrating how custom CSS classes can visually differentiate code for warnings, tips, errors, and more. ↩

---

1. ↩