Main Title

Class subtitle

07/09/2025 - 18/09/2025

Tom Planche
2025-2026
Class name

Table des matières

I - Main title 3

I. I - Maths 3

I. I .I - `#definition` 3

I. I .II - `#example` 3

I. I .III - `#theorem` 3

I. I. III. 1 - With `title` 3

I. I. III. 2 - Without `title` 3

I. I .IV - `ar` 4

I. II - Subtitle 4

I. II .I - Subsubtitle 4

I - Main title

I. I - Maths

For my maths class, I made these things:

I. I .I - `#definition`

Definition 1.1. (Linéarité):
On dit que φ est linéaire (homomorphisme) si:

$$\varphi(\lambda_1 X_1 + \lambda_2 X_2 + ... + \lambda_n X_n) = \lambda_1 \varphi(X_1) + \lambda_2 \varphi(X_2) + ... + \lambda_n \varphi(X_n) \tag{1.1.1.1}$$

I. I .II - `#example`

Example 1.1. (Example title): Basic text.
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magnam aliquam quaerat.

$$\begin{aligned} \varphi(0,0,0) = (0,0) = 0_{\mathbb{R}^2} \\ \varphi(\alpha X_1 + \beta X_2) \overset{?}{=} \alpha \varphi(X_1) + \beta \varphi(X_2) \end{aligned} \tag{1.1.2.2}$$

I. I .III - `#theorem`

I. I. III. 1 - With `title`

Theorem 1.1. (Théorème de Stokes):
Soit M une variété différentielle à bord, orientée de dimension n, et ω une $(n-1)$-forme différentielle à support compact sur M de classe C_1.
Alors, on a :

$$\int_M d\omega = \int_{\{\partial M\}} i^* \omega \tag{1.1.3.3}$$

où d désigne la dérivée extérieure, ∂M le bord de M, muni de l'orientation induite,
et $i^* \omega = \omega \mid_{\{\partial M\}}$ la restriction de ω à ∂M.

I. I. III. 2 - Without `title`

Theorem 1.2.
Soit E un espace vectoriel de dimension finie, F un sous-espace vectoriel de E et $B = (X_1, X_2, ..., X_n)$ une base de F.
Alors, il existe une base $\left(X_1, X_2, ..., X_n, X_{\{n+1\}}, ..., X_m\right)$ de E telle que $(X_1, X_2, ..., X_n)$ soit une base de F.

I. I .IV - `ar`

For vectors, I use `ar(X)` and it gives $\vec{X}$.

I. II - Subtitle

I. II .I - Subsubtitle

Custom Block

> Custom Blockquote

`Basic inline raw text`

This code block uses `#code()` macro.

src/string_utils.rs

```rust
/// Extension traits and utilities for string manipulation
///
/// This module provides additional functionality for working with strings,
/// including title case conversion and other string transformations.
use std::string::String;

/// Trait that adds title case functionality to String and &str types
pub trait TitleCase {
    /// Converts the string to title case where each word starts with an uppercase letter
    /// and the rest are lowercase
    ///
    fn to_title_case(&self) -> String;
}

impl TitleCase for str {
    fn to_title_case(&self) -> String {
        self.split(|c: char| c.is_whitespace() || c == '_' || c == '-')
            .filter(|s| !s.is_empty())
            .map(|word| {
                // If the word is all uppercase and longer than 1 character, preserve it
                if word.chars().all(|c| c.is_uppercase()) && word.len() > 1 {
                    word.to_string()
                } else {
                    let mut chars = word.chars();
                    match chars.next() {
                        None => String::new(),
                        Some(first) => {
                            let first_upper = first.to_uppercase().collect::<String>();
                            let rest_lower = chars.as_str().to_lowercase();
                            format!("{}{}", first_upper, rest_lower)
                        }
                    }
                }
            })
            .collect::<Vec<String>>()
            .join(" ")
    }
}

impl TitleCase for String {
    fn to_title_case(&self) -> String {
        self.as_str().to_title_case()
    }
}

#[cfg(test)]
mod tests {
    use super::*;

    #[test]
    fn test_title_case_str() {
        assert_eq!("hello world".to_title_case(), "Hello World");
        assert_eq!("HASH_TABLE".to_title_case(), "HASH TABLE");
        assert_eq!("dynamic-programming".to_title_case(), "Dynamic Programming");
        assert_eq!("BFS".to_title_case(), "BFS");
        assert_eq!("two-sum".to_title_case(), "Two Sum");
        assert_eq!("binary_search_tree".to_title_case(), "Binary Search Tree");
        assert_eq!("   spaced   words   ".to_title_case(), "Spaced Words");
        assert_eq!("".to_title_case(), "");
    }
}
```