Main Title

Class subtitle

04/09/2024 - 22/11/2024

Table des matières

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:
> $$\varphi(\lambda_1 X_1 + \lambda_2 X_2 + ... + \lambda_n X_n) = \lambda_1 \varphi(X_1) + \lambda_2 \varphi(X_2) + ... + \lambda_n \varphi(X_n)$$

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.
> $$\varphi(0,0,0) = (0,0) = 0_{\mathbb{R}^2}$$
> $$\varphi(\alpha X_1 + \beta X_2) \overset{?}{=} \alpha \varphi(X_1) + \beta \varphi(X_2)$$

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 :
> $$\int_M d\omega = \int_{\{\partial M\}} i^* \omega$$
> 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

<div style="text-align:center">

Custom Block

</div>

> Custom Blockquote

`Basic inline raw text`

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

```rust
1  // src/string_utils.rs
2  /// Extension traits and utilities for string manipulation
3  ///
4  /// This module provides additional functionality for working with strings,
5  /// including title case conversion and other string transformations.
6  use std::string::String;
7
8  /// Trait that adds title case functionality to String and &str types
9  pub trait TitleCase {
10     /// Converts the string to title case where each word starts with an uppercase letter
11     /// and the rest are lowercase
12     ///
13     fn to_title_case(&self) -> String;
14 }
15
16 impl TitleCase for str {
17     fn to_title_case(&self) -> String {
18         self.split(|c: char| c.is_whitespace() || c == '_' || c == '-')
19             .filter(|s| !s.is_empty())
20             .map(|word| {
21                 // If the word is all uppercase and longer than 1 character, preserve it
22                 if word.chars().all(|c| c.is_uppercase()) && word.len() > 1 {
23                     word.to_string()
24                 } else {
25                     let mut chars = word.chars();
26                     match chars.next() {
27                         None => String::new(),
28                         Some(first) => {
29                             let first_upper = first.to_uppercase().collect::<String>();
30                             let rest_lower = chars.as_str().to_lowercase();
31                             format!("{}{}", first_upper, rest_lower)
32                         }
33                     }
34                 }
35             })
36             .collect::<Vec<String>>()
37             .join(" ")
38     }
39 }
40
41 impl TitleCase for String {
42     fn to_title_case(&self) -> String {
```

```
43          self.as_str().to_title_case()
44      }
45 }
46
47 #[cfg(test)]
48 mod tests {
49     use super::*;
50
51     #[test]
52     fn test_title_case_str() {
53         assert_eq!("hello world".to_title_case(), "Hello World");
54         assert_eq!("HASH_TABLE".to_title_case(), "HASH TABLE");
55         assert_eq!("dynamic-programming".to_title_case(), "Dynamic Programming");
56         assert_eq!("BFS".to_title_case(), "BFS");
57         assert_eq!("two-sum".to_title_case(), "Two Sum");
58         assert_eq!("binary_search_tree".to_title_case(), "Binary Search Tree");
59         assert_eq!("   spaced   words   ".to_title_case(), "Spaced Words");
60         assert_eq!("".to_title_case(), "");
61     }
62 }
```