



Chancellor CLI - Markdown-Based L&T Materials

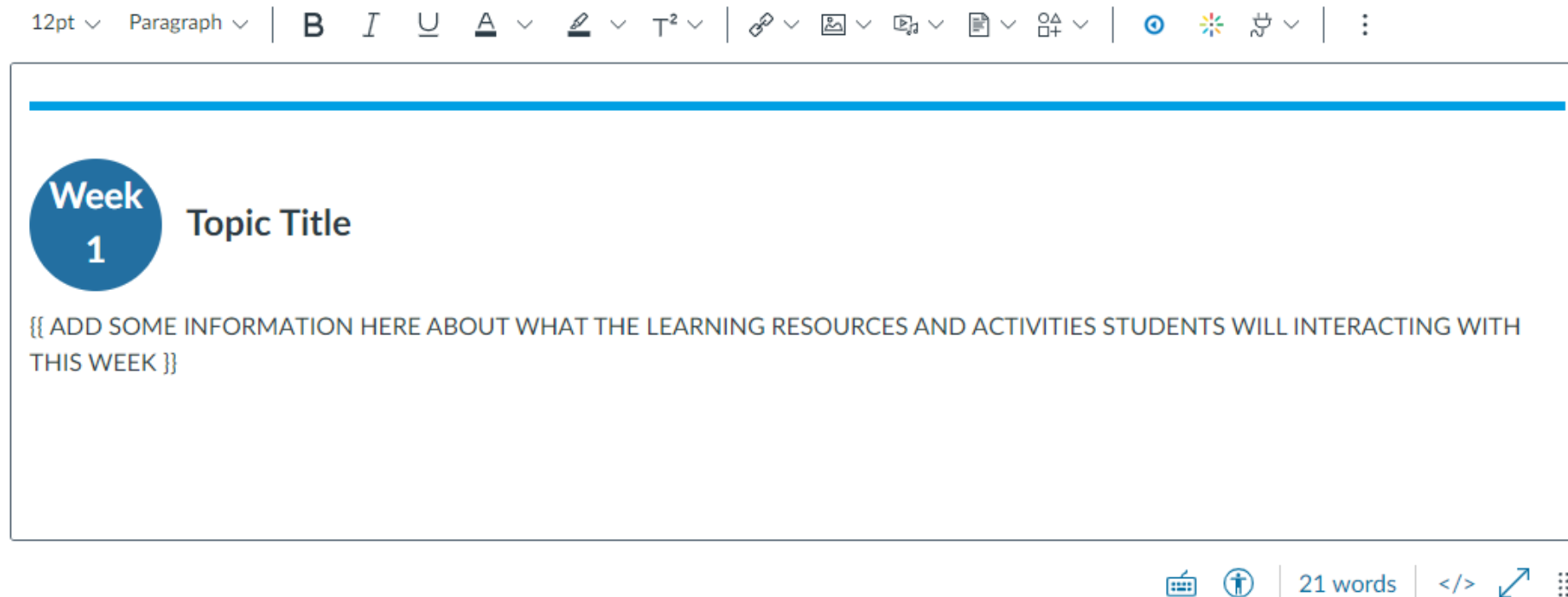
By: **Dan Tran**, under the supervision of:

- **Dr Jake Bradford** - Project Lead
- **Associate Professor Gentry White** - Project Member

Faculty of Science

Canvas LMS

The **Canvas Learning Management System (LMS)** allows the creation of online learning materials through a rich text editor with a WYSIWYG interface.



The Problem

While convenient for simple tasks like fixing typos, or uploading a single file, it has limitations for more **complex tasks** like:

- Creating a whole unit with **hundreds of pages** and **files**
- Curating a **consistent look and feel** across all materials
- **Updating** all materials for a new semester
 - E.g., update all occurrences of "2023" to "2024"
- And more...

Resulting in **tedious, error-prone, and time-consuming** work.

An inspiration - Markdown

Markdown is a lightweight markup language with plain text formatting syntax.

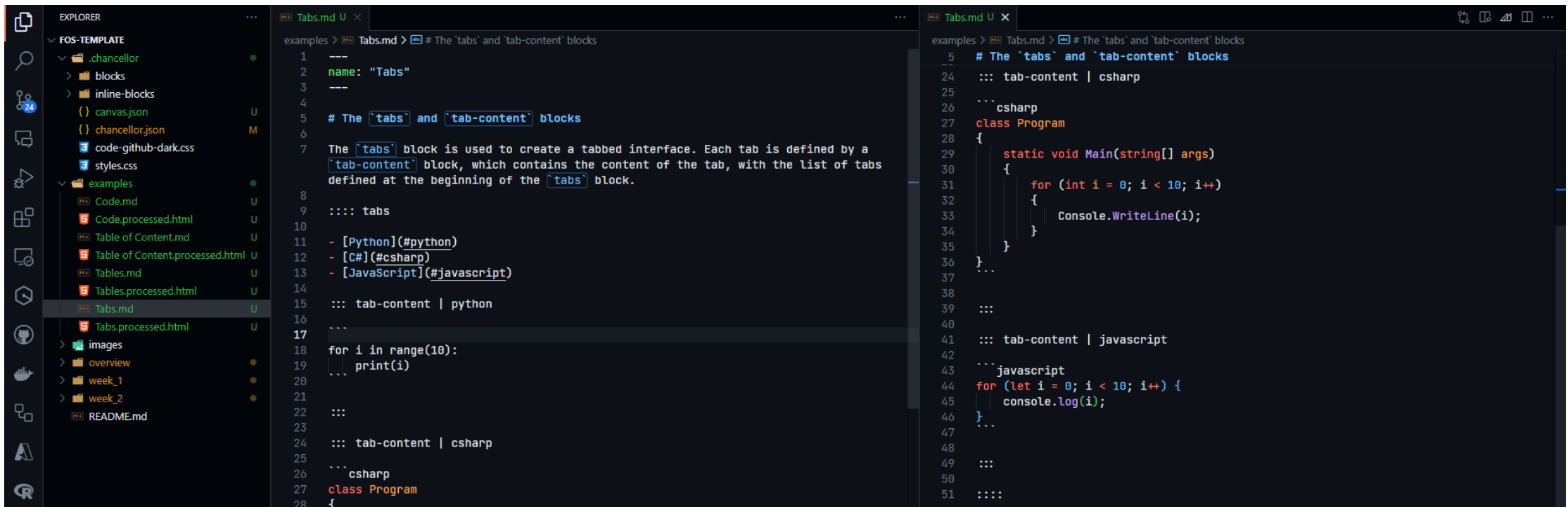
```
<div style="display: flex; gap: 1rem"> <div style="flex: 0.5">
```

```
**Bold text** and *italic text*  
- List item 1  
- List item 2  
```csharp  
class Program {
 static void Main() {
 Console.WriteLine("Hello, World!");
 }
}
```
```

```
</div> <small style="flex: 0.35; overflow-y: scroll">
```

The Solution - Chancellor

Write **Markdown** files with your favourite text editor (e.g., VS Code).



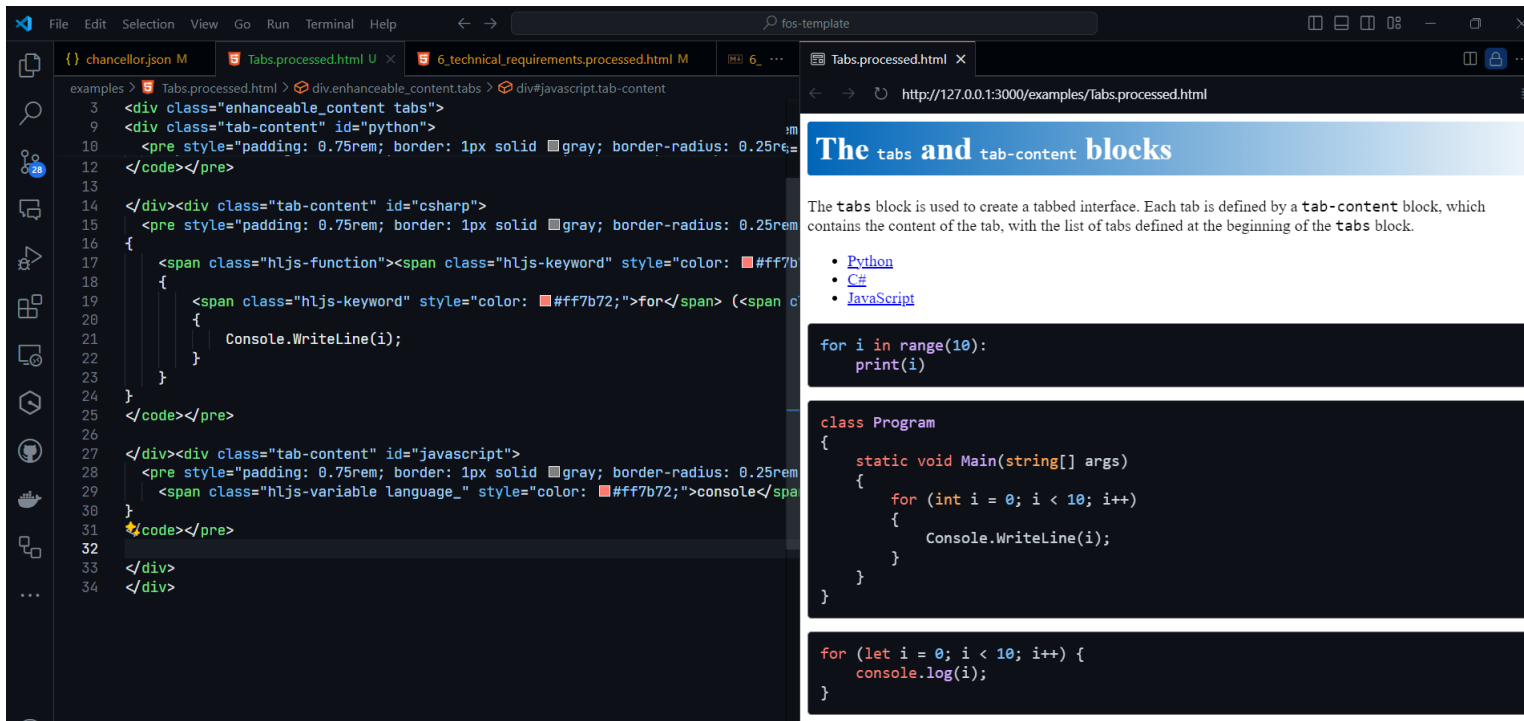
The screenshot displays the Visual Studio Code editor interface. On the left, the Explorer sidebar shows a project structure with folders like '.chancellor', 'blocks', 'inline-blocks', and 'examples'. The 'examples' folder is expanded, showing files including 'Code.md', 'Code.processed.html', 'Table of Content.md', 'Table of Content.processed.html', 'Tables.md', 'Tables.processed.html', 'Tabs.md', and 'Tabs.processed.html'. The 'Tabs.md' file is selected and open in the editor. The editor shows the Markdown content of 'Tabs.md', which includes a title '# The `tabs` and `tab-content` blocks', a description of the 'tabs' block, and a list of tabs: '[Python](#python)', '[C#](#csharp)', and '[JavaScript](#javascript)'. Below the list, there are three 'tab-content' blocks: one for 'python' containing a Python loop, one for 'csharp' containing a C# class, and one for 'javascript' containing a JavaScript loop. The right side of the editor shows the processed HTML output of the same file, where the Markdown syntax has been converted into HTML, including the 'tabs' block structure and the content of each tab.

```
1 ---
2 name: "Tabs"
3 ---
4
5 # The `tabs` and `tab-content` blocks
6
7 The `tabs` block is used to create a tabbed interface. Each tab is defined by a
8 `tab-content` block, which contains the content of the tab, with the list of tabs
9 defined at the beginning of the `tabs` block.
10
11 ::: tabs
12 - [Python](#python)
13 - [C#](#csharp)
14 - [JavaScript](#javascript)
15
16 ::: tab-content | python
17
18 for i in range(10):
19     print(i)
20
21
22
23
24 ::: tab-content | csharp
25
26 csharp
27 class Program
28 {
```

```
5 # The `tabs` and `tab-content` blocks
24 ::: tab-content | csharp
25
26 csharp
27 class Program
28 {
29     static void Main(string[] args)
30     {
31         for (int i = 0; i < 10; i++)
32         {
33             Console.WriteLine(i);
34         }
35     }
36 }
37
38
39
40
41 ::: tab-content | javascript
42
43 javascript
44 for (let i = 0; i < 10; i++) {
45     console.log(i);
46 }
47
48
49
50
51
```

The Solution - Chancellor

chancellor run render converts Markdown files to HTML.



The Solution - Chancellor

chancellor run canvas-deploy to upload to Canvas.

Tabs

The `tabs` and `tab-content` blocks

The `tabs` block is used to create a tabbed interface. Each tab is defined by a `tab-content` block, which contains the content of the tab, with the list of tabs defined at the beginning of the `tabs` block.

[Python](#) [C#](#) [JavaScript](#)

```
class Program
{
    static void Main(string[] args)
    {
        for (int i = 0; i < 10; i++)
        {
            Console.WriteLine(i);
        }
    }
}
```

◀ Previous

Alpha Release of the `chancellor-cli` tool

Deployed to `npm chancellor-cli` for easy installation.

- **Render** Markdown to HTML
 - Custom blocks
 - Styling with CSS
- **Deploy** to Canvas
 - Update existing pages or create new ones
 - Organise pages into modules
 - Automatically update files and links to other pages

Find out more at <https://github.com/Chancellor-LMS/chancellor-cli.git>

Demo & Showcase

- **Demo:** Deploying from the FoS template
- **Showcase:** Experimented on a few units to various degrees:
 - **CAB302** - Software Development
 - **EGD103** - Computing and Data for Engineers (QUT College)
 - **CAB201** - Programming Principles
 - **CAB301** - Algorithm and Complexity

Future Work

We are looking for **feedback** on the tool and **feature requests** via GitHub issues:

<https://github.com/Chancellor-LMS/chancellor-cli/issues>

A template is available at:

<https://github.com/Chancellor-LMS/fos-template>

Otherwise:

- **Graphical User Interface (GUI)** built on top of the CLI
- Using **Git** for version control
- **Documentation** and **user guides**
- (If time permits) Render to PDF, generate static site (GitHub Pages)

TEQSA Provider ID PRV12079 Australian University | CRICOS No. 92133