Doc As Code Example

Table of contents

Opening

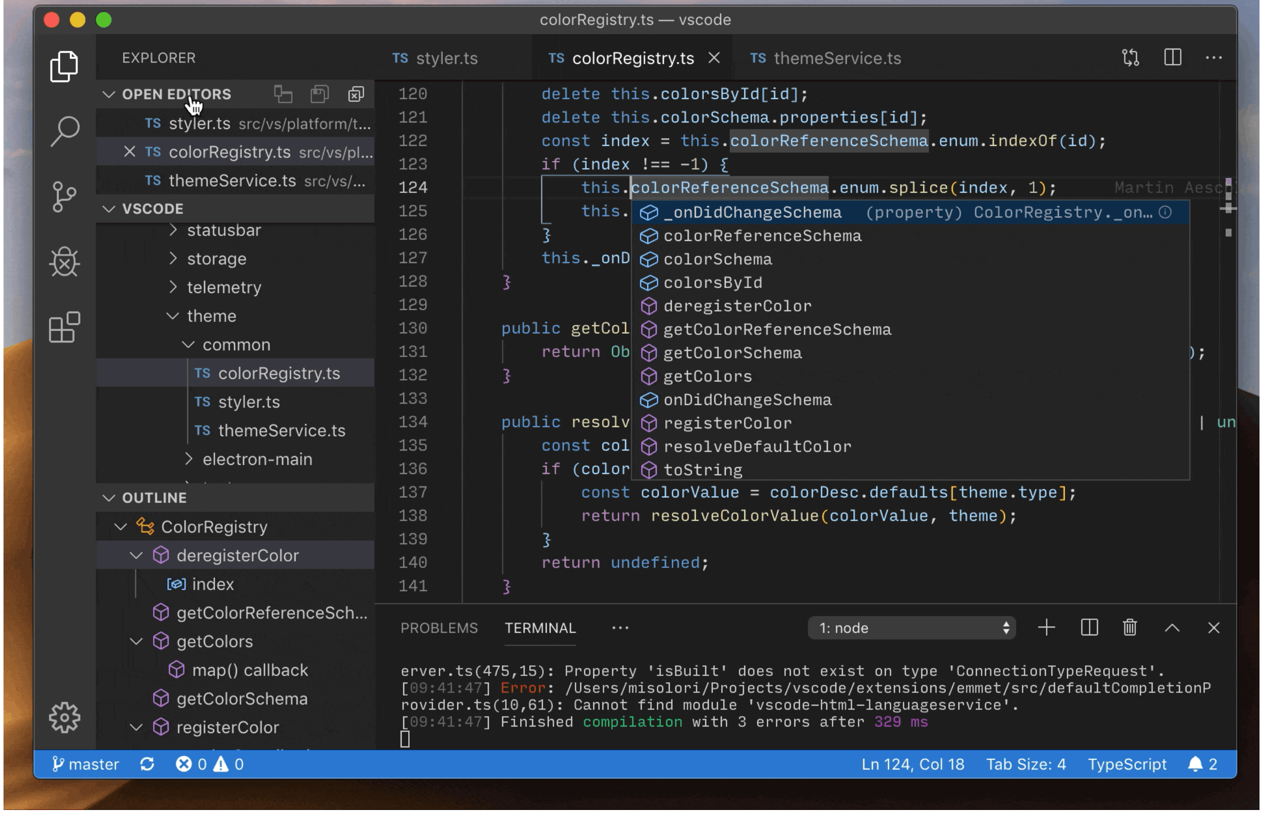
Docs as Code Documentation as Code (Docs as Code) refers to a philosophy that you should be writing documentation with the same tools as code:

- Issue Trackers
- Version Control
- Markdown
- Code Reviews
- Automated Tests
- This means following the same workflows as development teams, and being integrated in the product team. It enables a culture where writers and developers both feel ownership of documentation, and work together to make it as good as possible.

Generally a Docs as Code approach gives you the following benefits:

1. Writers integrate better with development teams
2. Developers will often write a first draft of documentation
3. You can block merging of new features if they don't include documentation, which incentivizes developers to write about features while they are fresh

In addition, there is an open source tool-chain which shows how the docs-as-code approach can be implemented



```
        delete this.colorsById[id];
        delete this.colorSchema.properties[id];
        const index = this.colorReferenceSchema.enum.indexOf(id);
        if (index !== -1) {
            this.colorReferenceSchema.enum.splice(index, 1);        Martin Aesc
            this.
        }
        this._onD
    }

    public getCol
        return Ob                                                            );
    }

    public resolv                                                            | un
        const col
        if (color
            const colorValue = colorDesc.defaults[theme.type];
            return resolveColorValue(colorValue, theme);
        }
        return undefined;
    }
```

```
erver.ts(475,15): Property 'isBuilt' does not exist on type 'ConnectionTypeRequest'.
[09:41:47] Error: /Users/misolori/Projects/vscode/extensions/emmet/src/defaultCompletionP
rovider.ts(10,61): Cannot find module 'vscode-html-languageservice'.
[09:41:47] Finished compilation with 3 errors after 329 ms
```

Pros and cons

As with any tech paradigm, docs like code has its pros and cons. Some of these are situational – docs like code is more useful for some companies than others, for instance.

Pros:

- It allows you to reuse both software and human resources: a software company almost certainly already has expertise and tools for version control, automation, modern build processes and so on. By using the same tools and processes, you reduce costs and increase the number of people at the company who can support the docs. This in turn helps avoid documentation becoming a silo. It helps you work closely with development: by sharing processes, and using tooling that devs are comfortable with, you open up the docs to easier collaboration. For example, if you need developers to review your docs, they're likely to already be comfortable doing this on GitHub as part of a pull request.
- The power of automation: docs like code tooling tends to be customizable, and support modern web technologies. This makes it possible to embed autogenerated docs, set up automated tests to check for everything from style adherence to whether screenshots are up to date, automatically deploy demos, and more.
- It allows you to write in a lightweight markup language. If you're a writer who finds this quicker than using a WYSIWYG, this is a big improvement to your writing experience.

Cons:

- It requires technical knowledge and increases the tech skills threshold for technical writers: creating and managing a docs like code webhelp usually requires familiarity with modern web development tools (especially static site generators), and with other common dev tooling such as git. Some tech writers may love diving into the command line and wrangling tooling. Others may be put off. This has implications for hiring and training.
- It works best for documenting software: if the documentation is writing directly about the code, it makes sense for the docs and code to be close to each other (this is where you can take advantage of autogeneration). If your company is a software company, you still benefit from reusing resources. The advantages of docs like code decrease for other types of companies. If you prefer a WYSIWYG writing experience, be aware that most static site generators don't include a WYSIWYG editor. You can add one using one of the (many) CMS products for static site generators, but this is additional setup effort.

sample

title: sample

How to write doc as code

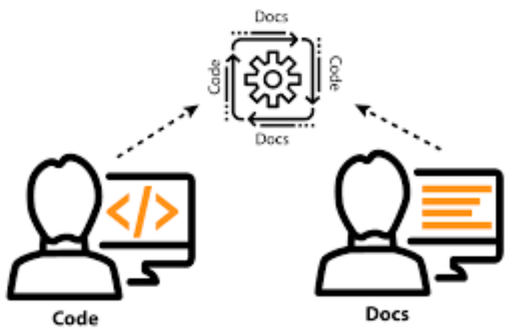
Body – CiscoSansTT Light 10pts in hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero eros et accumsan et iusto odio dignissim qui blandit praesent luptatum zzril delenit duis dolore te feugait nulla facilisi. Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam volutpat. Eum iriure dolor in hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero eros et accumsan et iusto odio dignissim qui blandit praesent luptatum zzril delenit duis dolore te feugait nulla facilisi. Lorem ipsum dolor sit amet, consectetuer adipiscing elit.

Git Workflow

forking workflow

The Forking Workflow is fundamentally different from other popular Git workflows. Instead of using a single server-side repository to act as the "central" codebase, it gives every developer their own server-side repository.

This means that each contributor has not one, but two Git repositories: a private local one and a public server-side one. The Forking Workflow is most often seen in public open-source projects.



The main advantage of the Forking Workflow is that contributions can be integrated without the need for everybody to push to a single central repository. Developers push to their own server-side repositories, and only the project maintainer can push to the official repository. This allows the maintainer to accept commits from any developer without giving them write access to the official codebase.

The following is a step-by-step example of this workflow.

1. A developer 'forks' an 'official' server-side repository. This creates their own server-side copy.
2. The new server-side copy is cloned to their local system.
3. A Git remote path for the 'official' repository is added to the local clone.
4. A new local feature branch is created.
5. The developer makes changes on the new branch.
6. New commits are created for the changes.
7. The branch gets pushed to the developer's own server-side copy.
8. The developer opens a pull request from the new branch to the 'official' repository.
9. The pull request gets approved for merge and is merged into the original server-side repository.

Team

Option	Org	Directory
Cole	Cloud-Ops	directory
Brett	GCC	directory
Jose	CPX-Sec	directory
Samone	CPX-Sec	directory
Tega	CPX-Sec	directory
Abena	CPX-Sec	directory
Martin	CPX-Sec	directory

Device Description

Intro - CiscoSansTT Regular 12/15 - Introduction paragraphs to be placed here. Lorem Ipsum is simply dummy text of the printing and typesetting industry. Eum iriure dolor in hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero eros et accumsan et iusto odio dignissim qui blandit praesent luptatum zzril delenit duis dolore te feugait nulla facilisi.

ToC_Subhead1CiscoSansTT Light 14pt

Body – CiscoSansTT Light 10pts in hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero eros et accumsan et iusto odio dignissim qui blandit praesent luptatum zzril delenit duis dolore te feugait nulla facilisi. Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam volutpat. Eum iriure dolor in hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero eros et accumsan et iusto odio dignissim qui blandit praesent luptatum zzril delenit duis dolore te feugait nulla facilisi. Lorem ipsum dolor sit amet, consectetuer adipiscing elit.