



The
ReadME
Project

PRESENTS

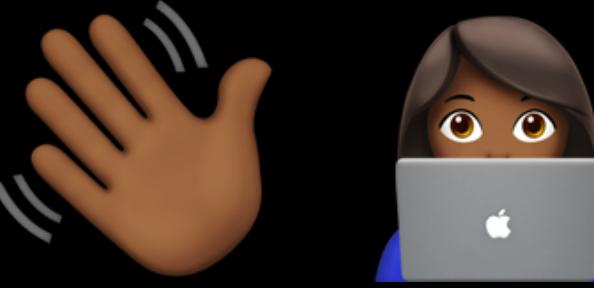
Using code as documentation to save time and share context



Monica Powell



M0nica/README.md



I am passionate about making contributing to open-source more approachable, creating technology to elevate people, and building community. Currently, I'm a **Senior Software Engineer at Newsela** building educational technology. In 2020, I was selected to be an inaugural **GitHub Star** based on my involvement in the tech community.



Code-based documentation can

- **Enhance traditional documentation** with templates that mirror decisions (i.e., preferred software, directory structure, naming conventions)
- **Reduce onboarding** time to:
 - set up a new machine
 - contribute to a new codebase



Maintain Configuration Files Remotely



Universe 2021 / Using Code as Documentation to Save Time and Share Context
@M0nica / October 2021

Where do remote files live?

Stored in the **cloud**, or a
remote git repository
(hosted on a site like
GitHub) *instead* of on your
local computer

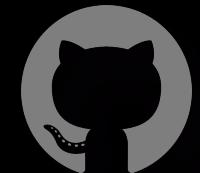


**How long would it
take you to set up a
new computer?**





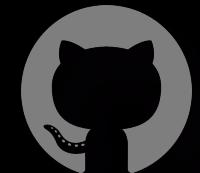
Days?



Universe 2021 / Using Code as Documentation to Save Time and Share Context
@M0nica / October 2021



Hours?



Universe 2021 / Using Code as Documentation to Save Time and Share Context
@M0nica / October 2021



Minutes?



Universe 2021 / Using Code as Documentation to Save Time and Share Context
@M0nica / October 2021

“

I estimate at least a three day's worth of work (downloading things and waiting for them to download 😴) if I have to install everything manually. But I was able to set my computer up in hours (automatically) thanks to dotfiles.”

Zell Liew, Migrating to a new Mac



Remote
configuration
files make setting
up lightning fast ⚡





Dotfiles

configuration files, traditionally hidden 🤫



```
5. bash
👑 Hold up, they don't love you like I love you
mina » Peppermint » ~/yoncé
» ls
}
```



Dotfiles

Configuration files
that **may be hidden**
by default 😇

- .gitignore
- .gitattributes
- .bashrc
- .bash_profile
- .zshrc
- .bash_history
- .zsh_history



gitignore

Keep your secrets
secret 🤫 and avoid
accidentally adding
large files to git 🐘

```
● ● ●  
# Logs  
logs  
yarn-error.log  
  
# Dependency directories  
node_modules/  
  
# dotenv environment variables file  
.env  
  
# generated Gatsby files  
.cache/  
public  
  
# Mac files  
.DS_Store
```



gitattributes

Define custom attributes for a git repository. Can be used to override the programming languages detected by GitHub's Linguist Library.

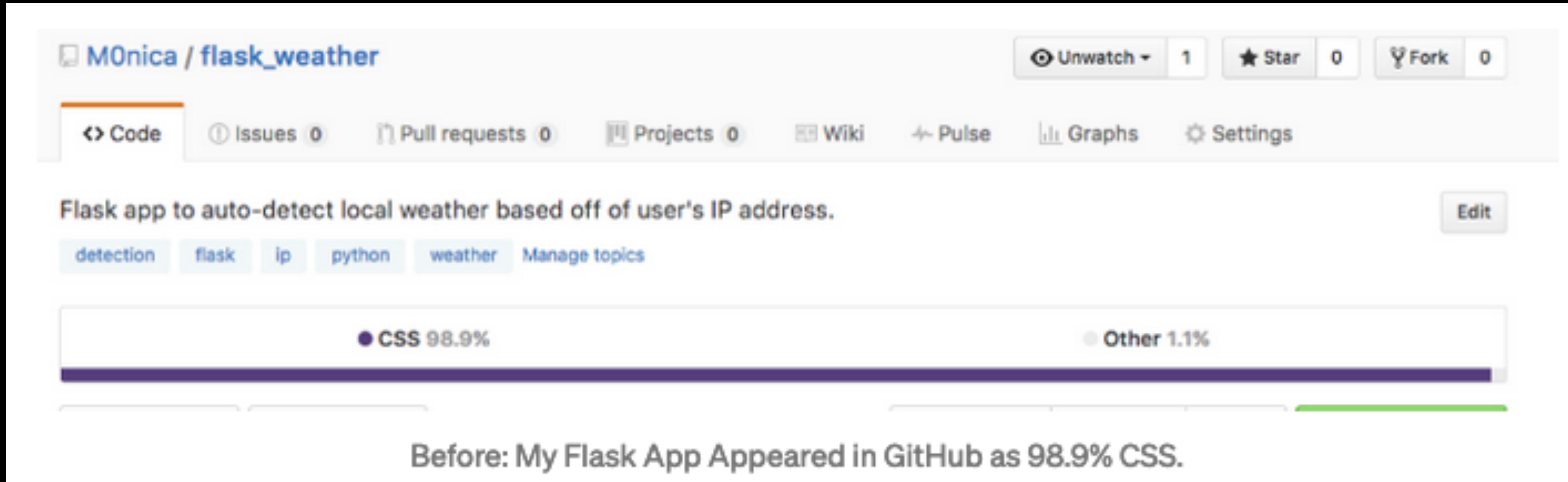


static/* linguist-vendored



gitattributes

98.9% CSS & 1.1% other



gitattributes

98.9% CSS & 1.1% other. →

56.2% Python and 43.8% HTML



bash_history && zsh_history

Stores recent commands and power autocomplete ✨



```
✓ TERMINAL
→ monica-site-playground git:(ohmyzsh) ✘ git clone https://github.com/zsh-users/zsh-autosuggestions ${ZSH_CUSTOM:-~/oh-my-zsh/custom}/plugins/zsh-autosuggestions
```

Ln 113, Col 121 Spaces: 2 UTF-8 LF Markdown React [D] Colorize Colorize: 0 variables



“

Brewfile: a Gemfile, but for Homebrew.”

Gabe Berke-Williams, Thoughtbot Blog



Install Apps via Homebrew

```
● ● ●

# Install Homebrew (if not installed)
echo "Installing Homebrew."
/bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"

# Installs Casks
brew tap homebrew/cask

## Apps to iNSTALL
brew install firefox
brew install adobe-creative-cloud
brew install github
brew install slack
brew install spotify
brew install visual-studio-code
brew install zoom

# Remove outdated versions from the cellar.
brew cleanup
```



Automate File Creation for Standardization



Universe 2021 / Using Code as Documentation to Save Time and Share Context
@M0nica / October 2021

**How long can you work on
making a routine task more
efficient before you're spending
more time than you save?**

How often do you do the task? Daily? Weekly?

How much time do you shave off? Seconds? Minutes?

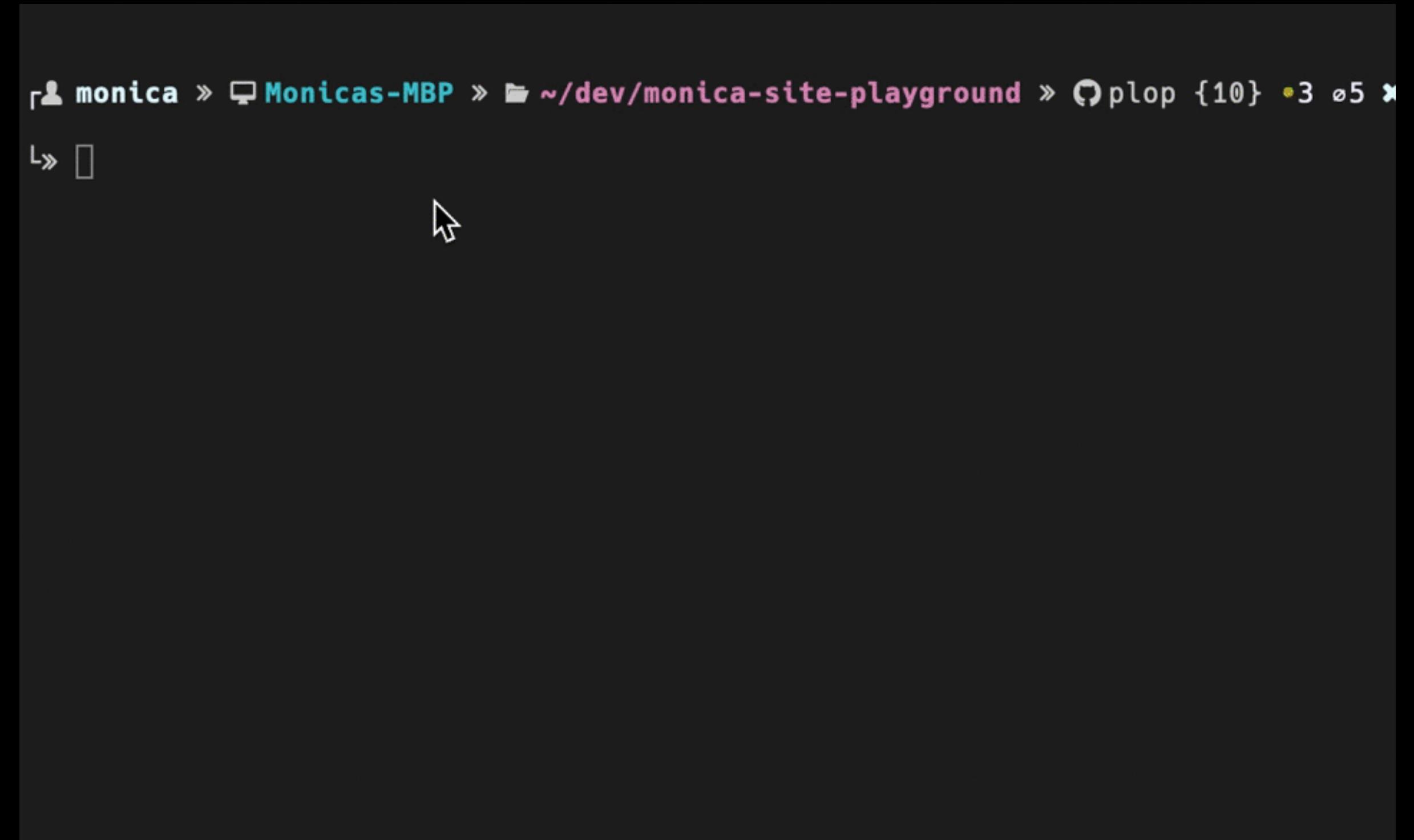


Scaffolding saves time

- Reduces the need to reference or copy/paste boilerplate code
- Automate the consistent, implementation of design patterns and best practices
- Reduces time to generate new projects, components and features



CLI to Generate Starter Files for New Features



```
─ monica » └── Monicas-MBP » └── ~/dev/monica-site-playground » ⌂ plop {10} •3 •5
```



PlopJS



Handlebars + InquirerJS



Universe 2021 / Using Code as Documentation to Save Time and Share Context
@M0nica / October 2021

Scaffolding helps answer common questions

- Where should **translations for this component** live?
- What's our **preferred testing approach**?
- What **files does this type of feature typically need**?
Does it need **its own README**?
- Should I use **camelCase**? **snake_case**? **kebab-case**?
PascalCase? **UPPERCASE_SNAKE_CASE**?



Using Plop in Chakra-UI

TERMINAL PORTS PROBLEMS OUTPUT

@M0nica → /workspaces/chakra-ui (main) \$

Live Share Jest: stopped | Jest-WS: 0 0 0 0



Using Plop in Chakra-UI



The screenshot shows a code editor interface with a dark theme. On the left is a sidebar with various icons for source control, changes, and other repository details. The main area displays a file named `GithubUniverse.test.tsx`. The code itself is a Jest test for a hook named `useGithubuniverse`. It includes a detailed comment block at the top with notes for contributors, specifying that tests should focus on component behavior and accessibility, and that snapshot tests should not be used. The code uses modern JavaScript syntax with imports from `@chakra-ui/test-utils` and `react`.

```
1  /**
2   *  Notes for Contributors:
3   *
4   * - Ensure you write tests for component behavior defined in the hook.
5   * - Ensure you write tests for the accessibility and interactions.
6   * - No snapshot tests for components please! 😊
7   *
8   * @see Testing-Guide https://chakra-ui.com/guides/component-guide#4-build-and-test
9  */
10
11 import { renderHook } from "@chakra-ui/test-utils";
12 import * as React from "react"; 7.8K (gzipped: 3.2K)
13 import { useGithubuniverse } from "../src";
14
15 describe("useGithubuniverse", () => {
16   test("it works", () => {
17     expect(true).toBeTruthy();
18   });
19});
```



Scaffolding works alongside traditional documentation

Generated starter files

Can answer **where**, **what** and **how** regarding code conventions

Traditional written documentation

Can provide more context on the **why** behind certain conventions



Documenting within a project



Proximity of Documentation to Code

documentation that lives in close proximity to the relevant code helps keep maintaining documentation top of mind as the software evolves.



Documentation within Codebase

- In-line comments to **explain the why** in conjunction with **meaningful variables and refactoring code** for understanding
- Type declarations **for languages that are not strongly typed**
- Using **TODO/FIXME** or other verbiage that can be **automatically parsed from AST via custom or third-party plugins** like Todo Tree or TODO Highlight



Thank you!

Learn more at github.com/readme/guides/code-as-documentation



[indigitalcolor](#)



[M0nica](#)



Universe 2021 / Using Code as Documentation to Save Time and Share Context
@M0nica / October 2021