Engineering Maintainable WordPress Websites

Toru Miki

WordCamp Taiwan 2025

Agenda

1. Introduction: Why Maintainability Matters
2. Principles of Plugin Selection
3. Separation of Concerns
4. Software Engineering Practices
5. AI as a Helper
6. Closing & Takeaways

01 1. Introduction & Setup

Why Maintainability Matters

It's an investment in the future of your project and your team

- **Future-Proofing**: Build sites that can easily adapt to future changes and requirements
- **Collaboration**: Enable smoother teamwork and **painless project handovers**
- **Audience**: This talk is for developers, agencies, and site maintainers

02 2. Plugin Selection

Principles of Plugin Selection

Look beyond the feature list to evaluate long-term viability

Responsively Maintained
Responsive authors, timely fixes (esp. security issues), and honest support – **not flashy but reliable**.

Prefer Single-Purpose
Avoid bloated "kitchen sink" plugins. Opt for **focused solutions** that do one thing well.

Secure (and scalable), well-written code
Built with security in mind, using clean practices that prevent fragile or unsafe code - **quality matters**.

Principles of Plugin Selection

Look beyond the feature list to evaluate long-term viability

Whatever you choose, remember this:

When you use a 3rd-party plugin, you are effectively handing off completely, its functionality and maintenance to someone else – 丸投げ or 完全丟給別人.

Remember, it's till your responsibility to handle its impact (updates, break, discontinued, etc).

We find that often 3rd-party plugins are the ones that causes issues in mid-to-long term maintenance support.

03

3. Logs… Logs!

Logs (Especially Error Logs)

Stop relying on guessworks!

Always keep logs accessible
They’re the first place to look when something goes wrong.

Error logs are gold
The tell you what actually happened – not what you think happened.

Make it part of your workflow
Check logs after deployments, plugin updates, or unexpected behavior.

03

3. Separation of Concerns

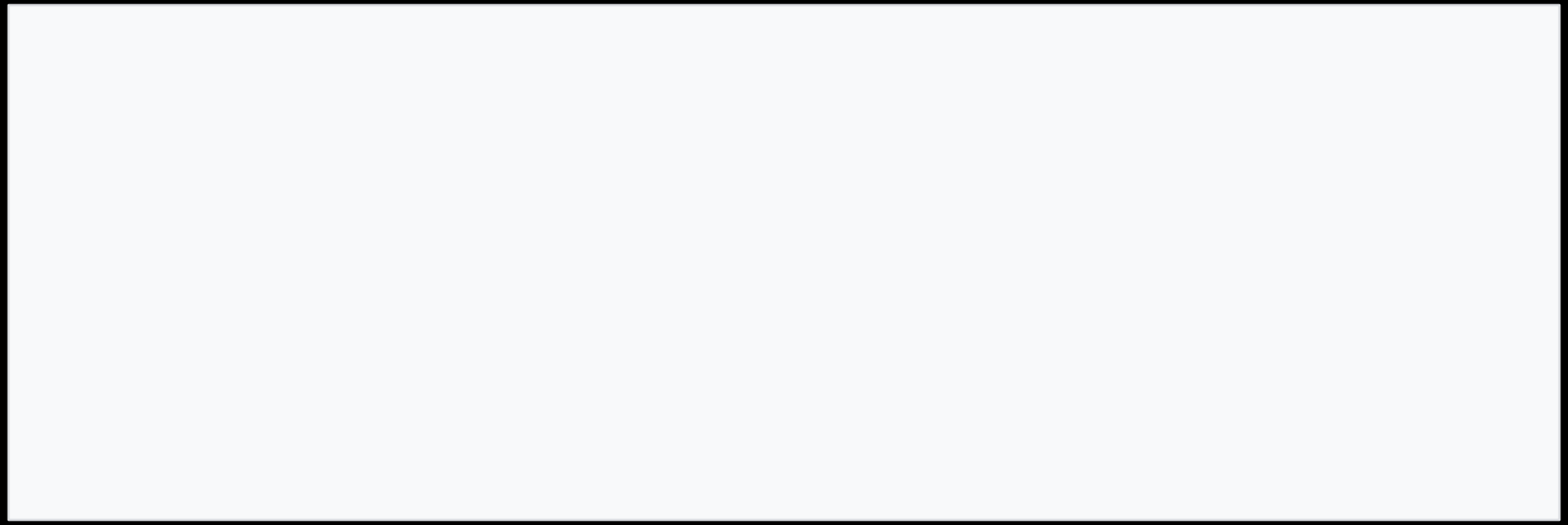
Separation of Concerns

A clear division of roles between themes and plugins

Good Practices	Bad Practice
Theme = **Presentation**	Functionality
Plugin = **Functionality**	Features which should not be tied to theme is coded in a theme All custom logics buried in `functions.php`
Custom features in dedicated, single-purpose plugins	Functionality is tightly coupled to a specific theme
Functionality is visible and discoverable	Hard to track where custom features live
	Fragile custom hacks instead of stable plugins

Separation of Concerns

Eg. Custom code in functions.php



04 4. Software Engineering Practices

Development Practices

Adopt tools and workflows from the broader software engineering world

Version Control	Collaboration	Documentation
Use **Git** to track every change and maintain a clear, revertible history	Use GitHub (or similar) for **Pull Request**-based workflows and code reviews	Write lightweight docs to ease future maintenance and project handovers
Static Analysis	**Testing**	**CI/CD**
Apply tools like PHPCS and ESLint to **enforce coding standards** automatically	Add tests where possible to prevent regressions and ensure stability	Set up automated pipelines for **consistent and safe deployments**

05

5. AI as a Helper

AI as a Helper, Not a Replacement

Leverage AI to accelerate repetitive tasks, but retain engineering oversight

- **AI excels at**: Generating boilerplate code, fixing lint errors, and drafting documentation
- **Humans are required for**: **Sound engineering judgment**, architecture decisions, and final approval
- Always review and test AI-generated code

06

6. Closing & Takeaways

Key Takeaways

- Maintainable WordPress leads to **sustainable business** and developer happiness
- A clean architecture (**Separation of Concerns**) is your most powerful tool
- Adopt modern development practices like Git and code reviews
- Start small. Adopting just **one or two new practices** makes a huge difference