A WordPress Perspective

About me

I am a WordPress and PHP specialist

I started my first projects in the late 90s.

I worked on projects with millions of page impressions. As technical lead I developed custom themes, plugins, optimized performance and planned coming features.

I am a frequent speaker at WordPress events, and founder and co-host of the Meetup in Dortmund.

Currently I work as Senior Web Developer, WordPress Expert and DX Engineer at Coding Pioneers in Iserlohn.

https://www.linkedin.com/in/christoph-daum/

Meet Sheldon

- He mostly worked alone
- He is confident of his skills
- He likes how WordPress is coded

Q: Coding Standards

Applying coding standards doesn't improve my code, it's just a waste of time.

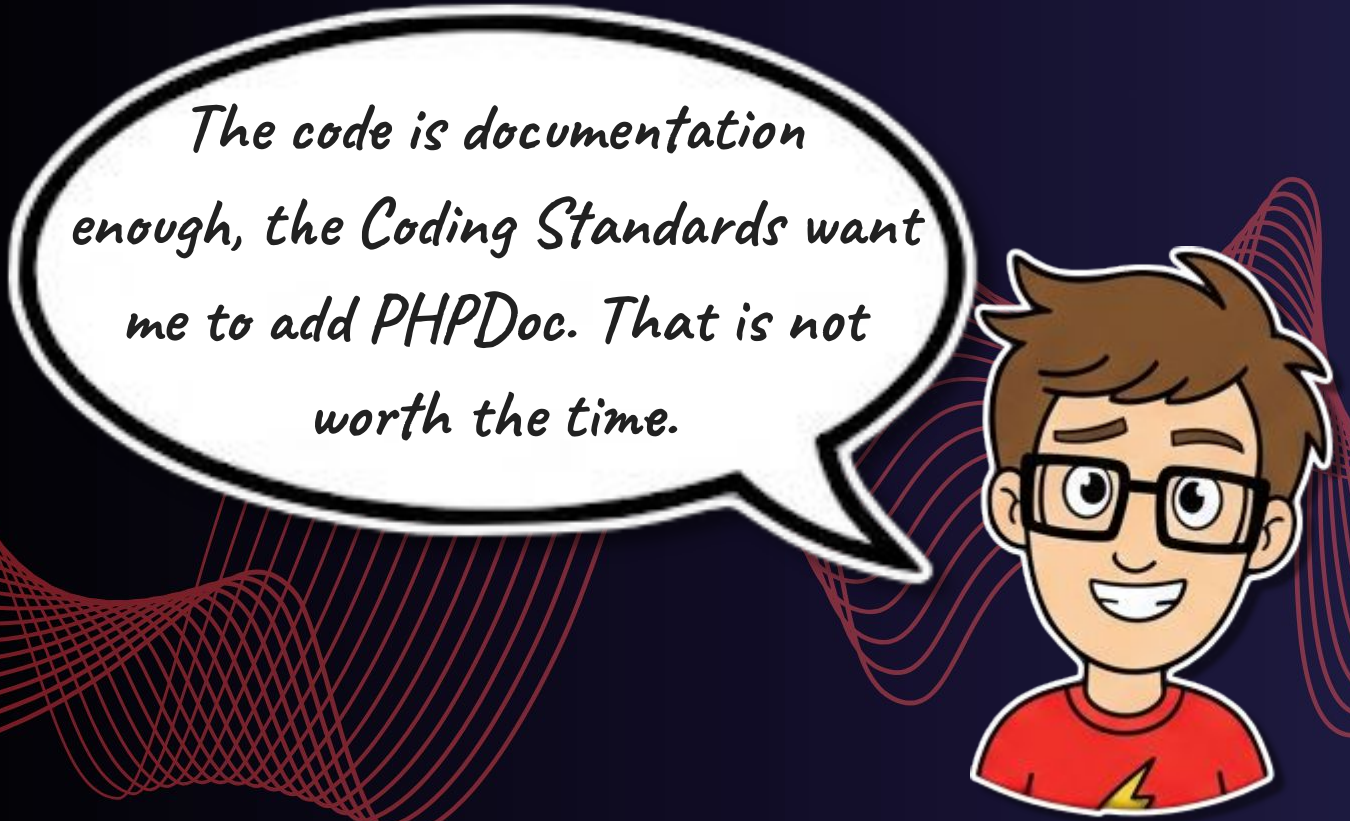
Q: Coding Standards

A: Coding Standards

Coding is not an Art - Coding is a Craft

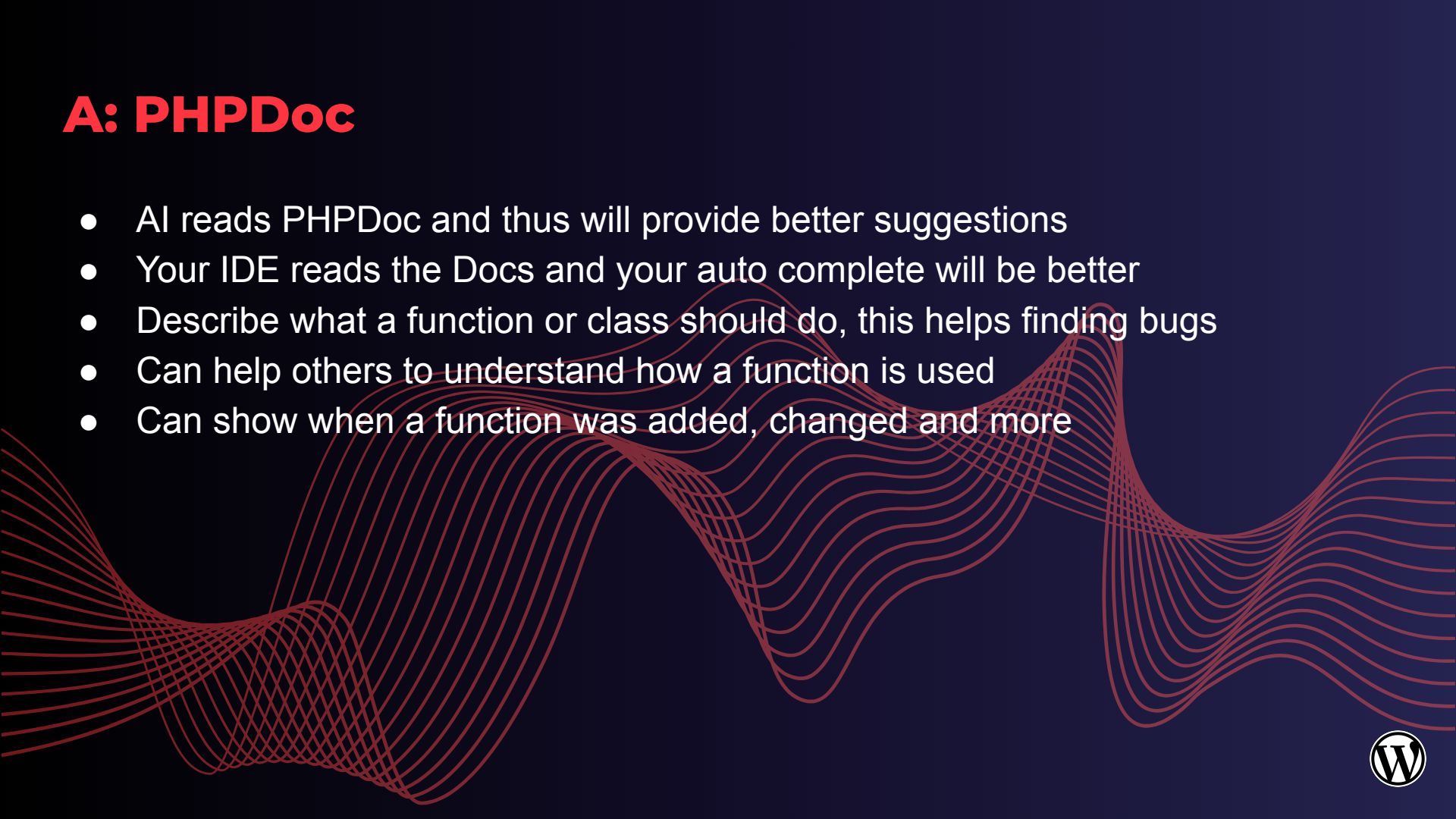
- Coding Standards help to work together
- Unified Styling (Usages of spaces, naming conventions etc.)
- Navigating through code will become easier
- Technical sniffs
  - Escaping
  - Proper preparation for i18n
  - Warning that meta queries are slow

Q: PHPDoc


The code is documentation enough, the Coding Standards want me to add PHPDoc. That is not worth the time.

A: PHPDoc

- AI reads PHPDoc and thus will provide better suggestions
- Your IDE reads the Docs and your auto complete will be better
- Describe what a function or class should do, this helps finding bugs
- Can help others to understand how a function is used
- Can show when a function was added, changed and more



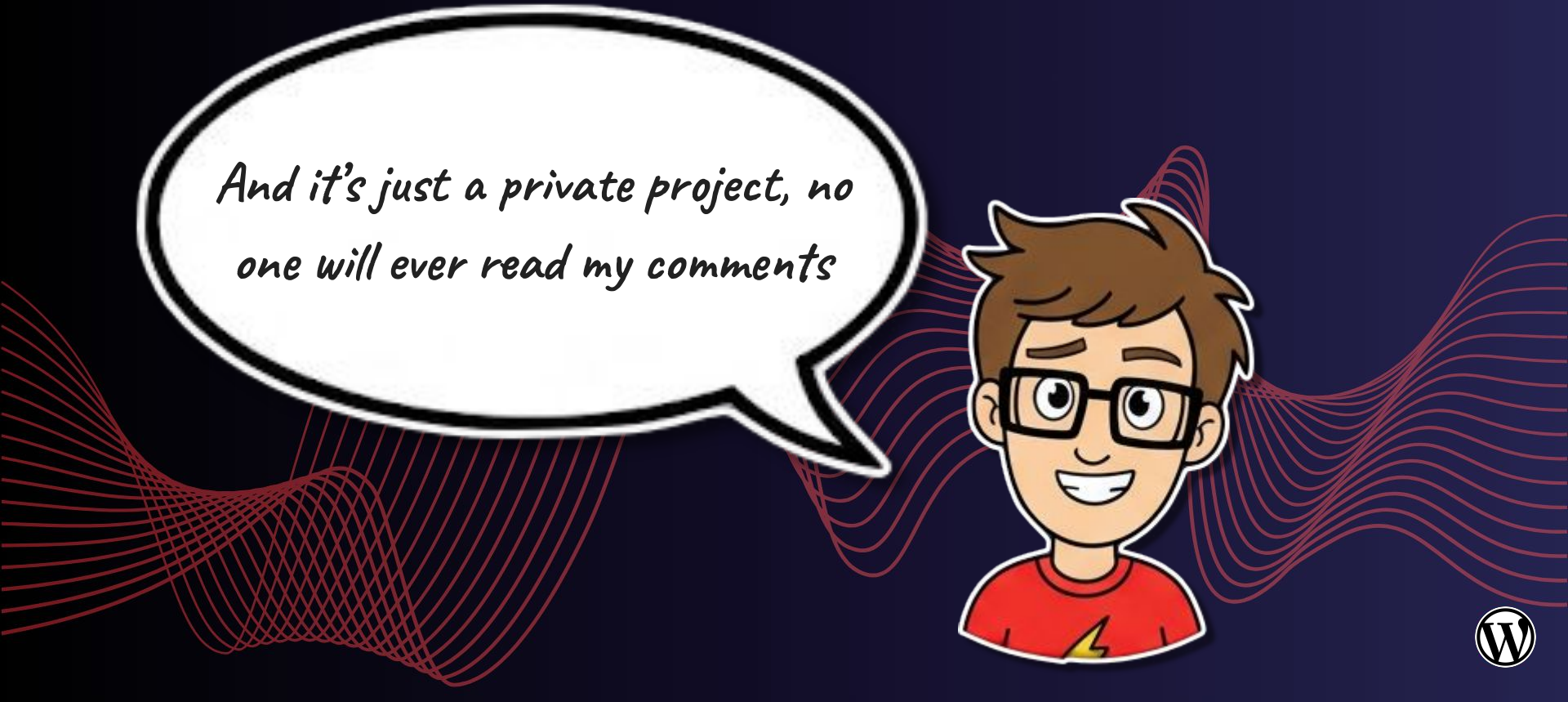
Q: Commented out code

A: Commented out code

- Always write when and why code was commented out
- If you forget to remove it, it will be kept forever
- In most cases, your git history is all you need
- Better write a comment with a hint to your git repository

Q: Professional Comments

Q: Professional Comments



A: Professional Comments

- Build a habit of writing professional comments
- Comments can be read by colleagues, clients, future employers
- Ranting, insulting or other offensive comments can cost your reputation
- Comments are not for venting, insulting and more
- If a comment is on a public github repository, everyone can read it

Found in the JavaScript of an actual project

```
// I'm very sorry. It had to be done. Quick, hard and dirty,
// that's how you like it, isn't it...
```

Q: Main Branch vs feature branches

A: Main Branch vs feature branches

Will cause problems if…

- you work with multiple people
- you work on several features at once
- you want to do major refactoring is basically impossible
- you want to have multiple stages of your website

Q: Pull Request and Code Review

A: Pull Request and Code Review

- Code Review is a main staple in software development
- Share and learn from each others
- 4 eyes principle for code
- Prevent logic errors
- Prevent Debug Code from being merged
- Fix Typos
- Gain confidence before deploying

Q: Pair Programming

If a second developer should look at my code, I heard about pair programming. What about that?

A: Pair Programming

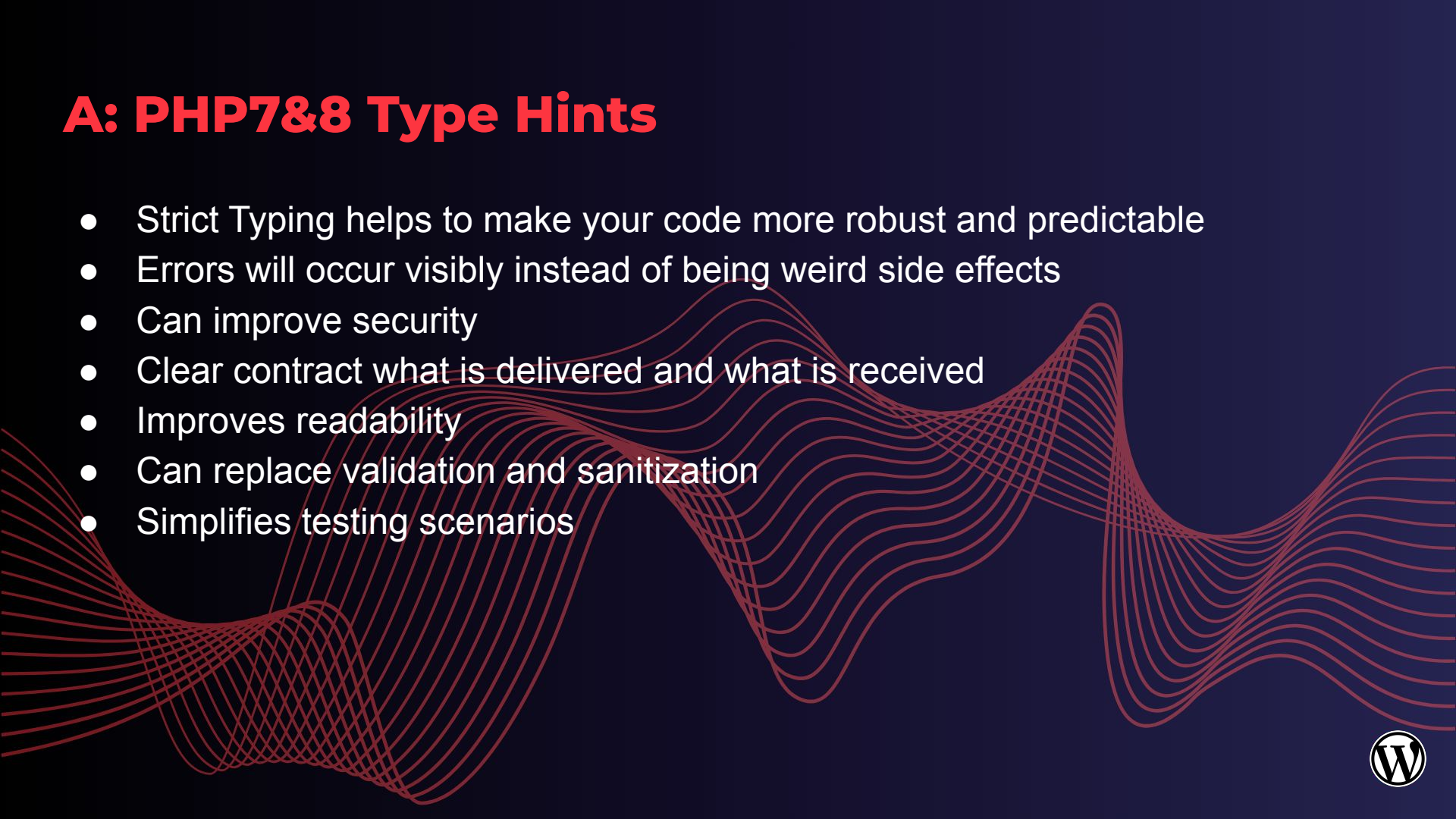
- Pair Programming CAN replace Code Review - depending on your teams rules
- for sharing knowledge
- for debugging
- for developing ideas
- for software architecture

Q: PHP7&8 Type Hints

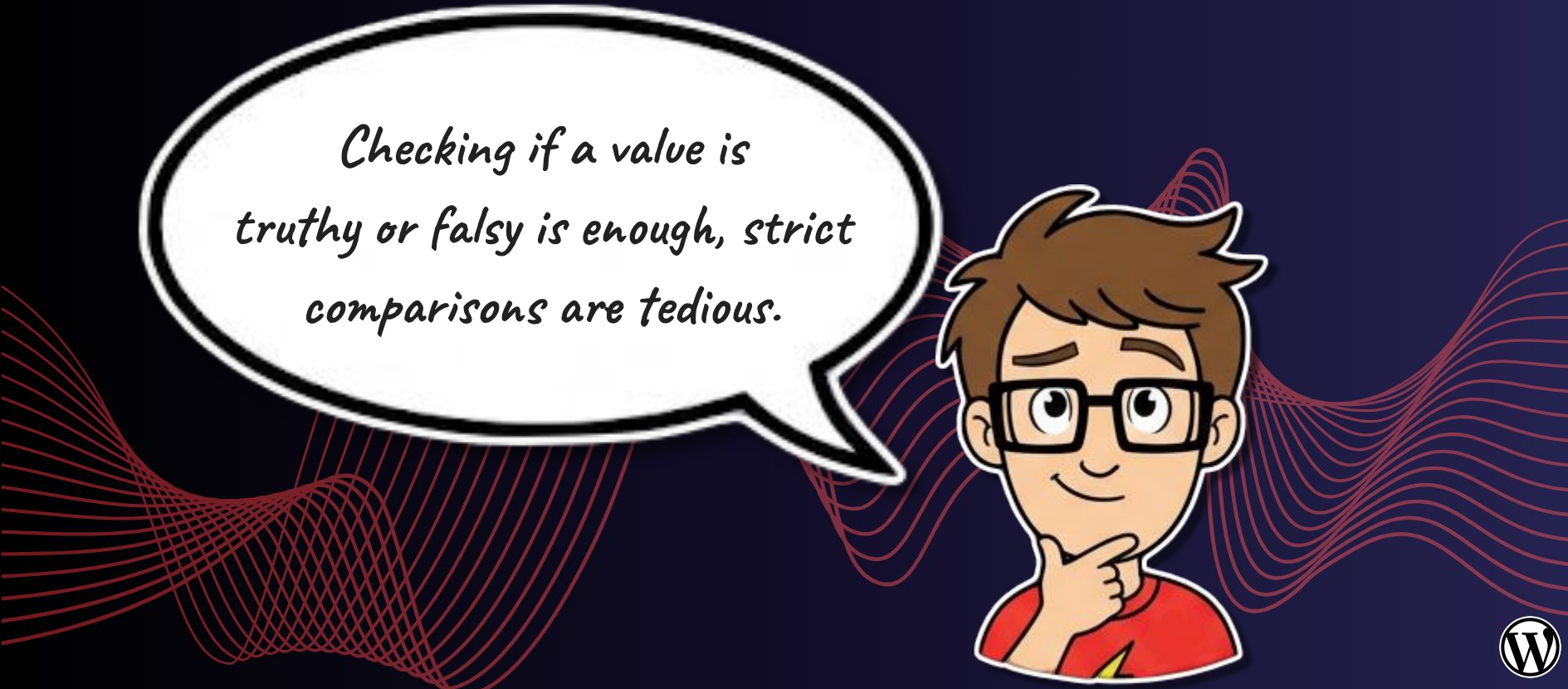
One of the strengths of PHP is being type less. I don't need to add types, that just slows me down.

A: PHP7&8 Type Hints

- Strict Typing helps to make your code more robust and predictable
- Errors will occur visibly instead of being weird side effects
- Can improve security
- Clear contract what is delivered and what is received
- Improves readability
- Can replace validation and sanitization
- Simplifies testing scenarios



Q: Strict comparisons and conditions



A: Strict comparisons and conditions

Strict comparison in conditions will prevent unpredicted behaviour

Instead of checking if something is truthy, you can check if contains the right type.

```php
if ( $post ) {
    echo $post->post_title;
}
```

Can result in a fatal error if $post is WP_Error

```php
if ( $post instanceof \WP_Post ) {
    echo $post->post_title;
}
```

Q: DRY - Don’t repeat yourself

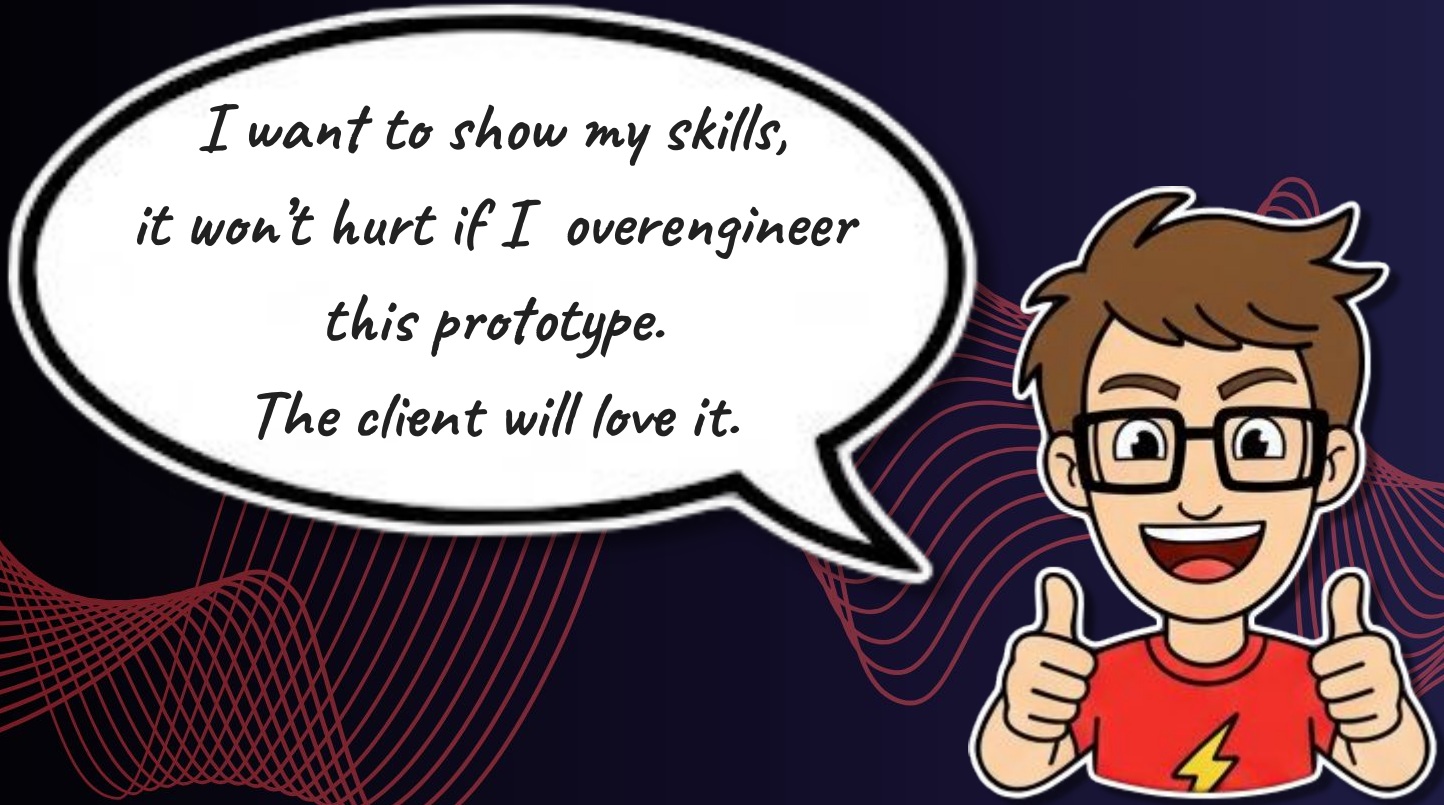
A: DRY - Don’t repeat yourself

Copy-pasting creates maintenance nightmares.

- ✅ Extract repeated logic
- ✅ Create reusable functions
- ✅ Share common validation
- ✅ Use inheritance/composition wisely

- ❌ Don't copy-paste code
- ❌ Don't repeat validation
- ❌ Don't duplicate queries
- ❌ Don't over-abstract simple logic
- ❌ Don't create unnecessary dependencies

Q: KISS - Keep it simple, stupid



A: KISS - Keep it simple stupid

✅ Choose simple solutions

✅ Avoid unnecessary complexity

✅ Write readable code

✅ Use familiar patterns

✅ Question every abstraction

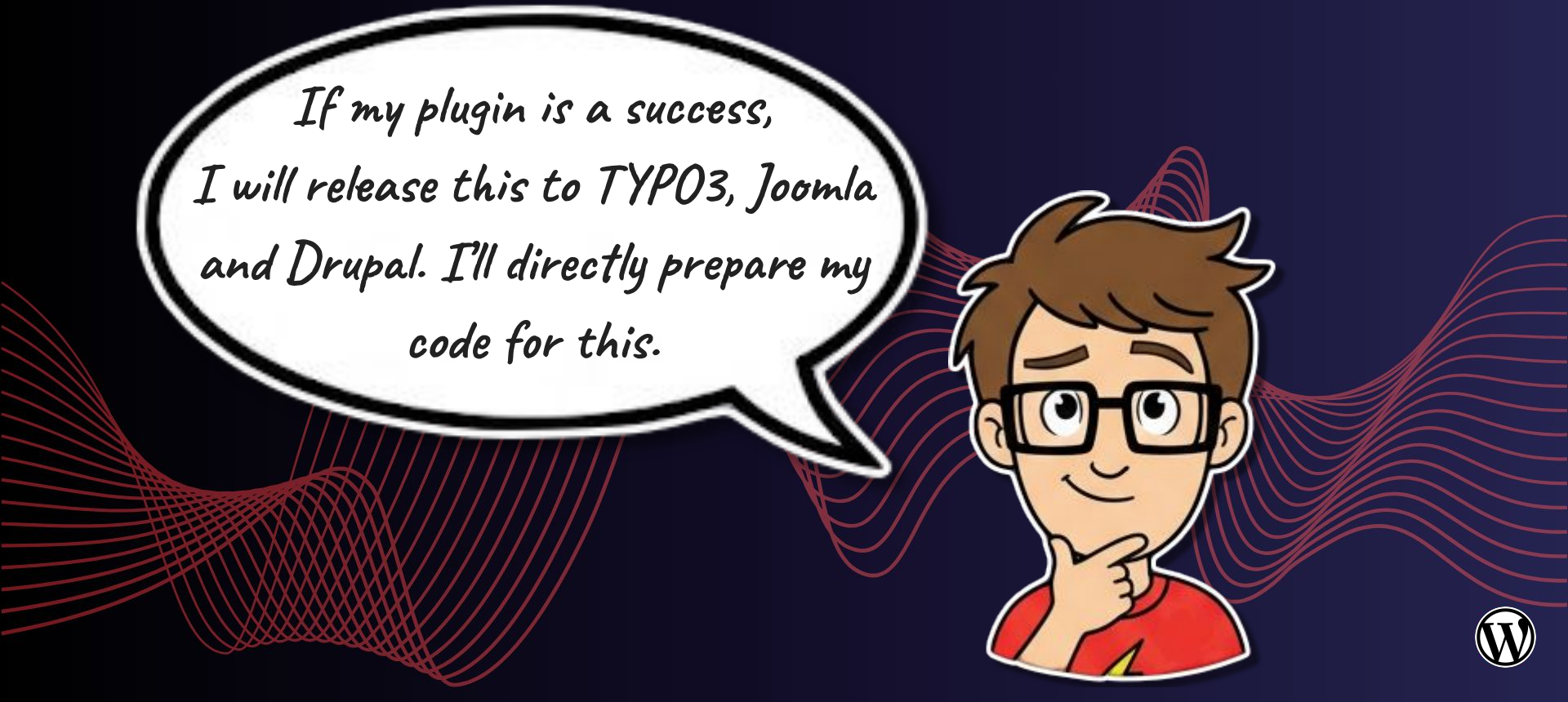
❌ Don't use design patterns for simple tasks

❌ Don't add layers without reason

❌ Don't make simple things complicated

❌ Don't prioritize "clever" over "clear

Q: YAGNI - You ain’t gonna need it



A: YAGNI - You ain’t gonna need it

- ✅ Build what you need NOW
- ✅ Add features WHEN requested
- ✅ Keep it simple
- ✅ Refactor when pattern emerges

- ❌ Don't build "just in case"
- ❌ Don't over-engineer
- ❌ Don't add speculative features
- ❌ Don't create unused abstractions
- ❌ Don't skip all planning

Q: Software Design Principles (SOLID)

Ok, but a function or class that will do everything is easier to read and you can follow the code.

A: Software Design Principles (SOLID)

- **S**ingle Responsibility
- **O**pen-closed principle
- **L**iskov substitution principle
- **I**nterface segregation principle
- **D**ependency inversion principle

https://en.wikipedia.org/wiki/SOLID

A: Single Responsibility

Each class and function should only have one responsibility

- Maintainability - Simple and well-defined code is easier to understand and modify
- Testability - It’s easier to test, especially with Unit Test
- Flexibility - Changes to one part won’t have side effects to other parts

A: Open-closed principle

Code should be open for extension but closed for modification

- Extensibility - new features can be added without modifying existing code
- Stability - reduces the risk of introducing bugs when making changes
- Flexibility - adapts to changing requirements more easily

A: Liskov substitution principle

Code must be able to use objects of derived classes without knowing it

- Polymorphism - Making code more flexible and reusable.
- Reliability - Subclasses adhere to the contract defined by the superclass.
- Predictability - Using the object of a subclass won't break the program

A: Interface segregation principle

Code should not depend upon interfaces that it does not use.

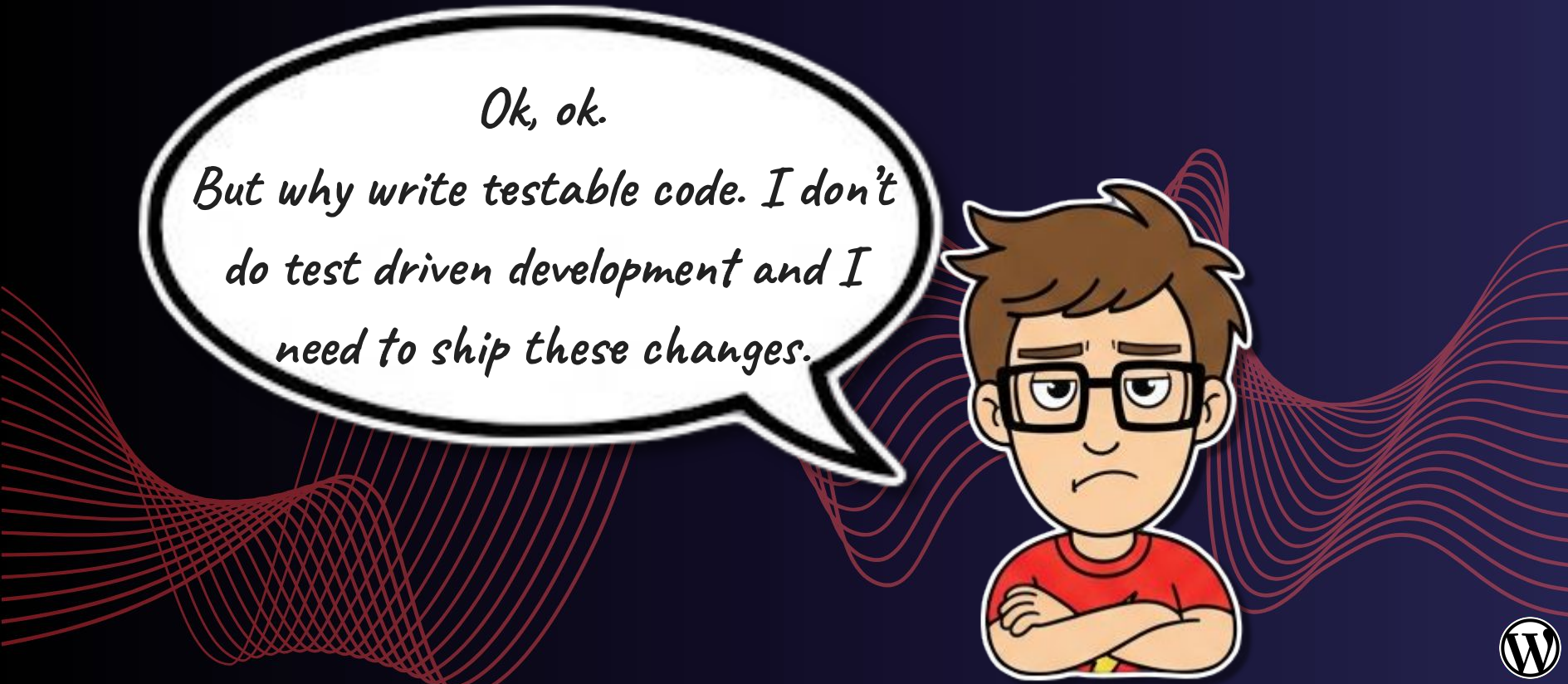
- Decoupling - Reduce dependencies between classes, make the code more modular and maintainable.
- Flexibility - Allow for more targeted implementations of interfaces.
- Avoids unnecessary dependencies - Don't have to depend on methods you don't use.

A: Dependency inversion principle

Depend upon abstractions, not concretes.

- Loose coupling - Reduce dependencies between modules, make the code more flexible and easier to test
- Flexibility - Enable changes to implementations without affecting clients
- Maintainability - Makes code easier to understand and modify

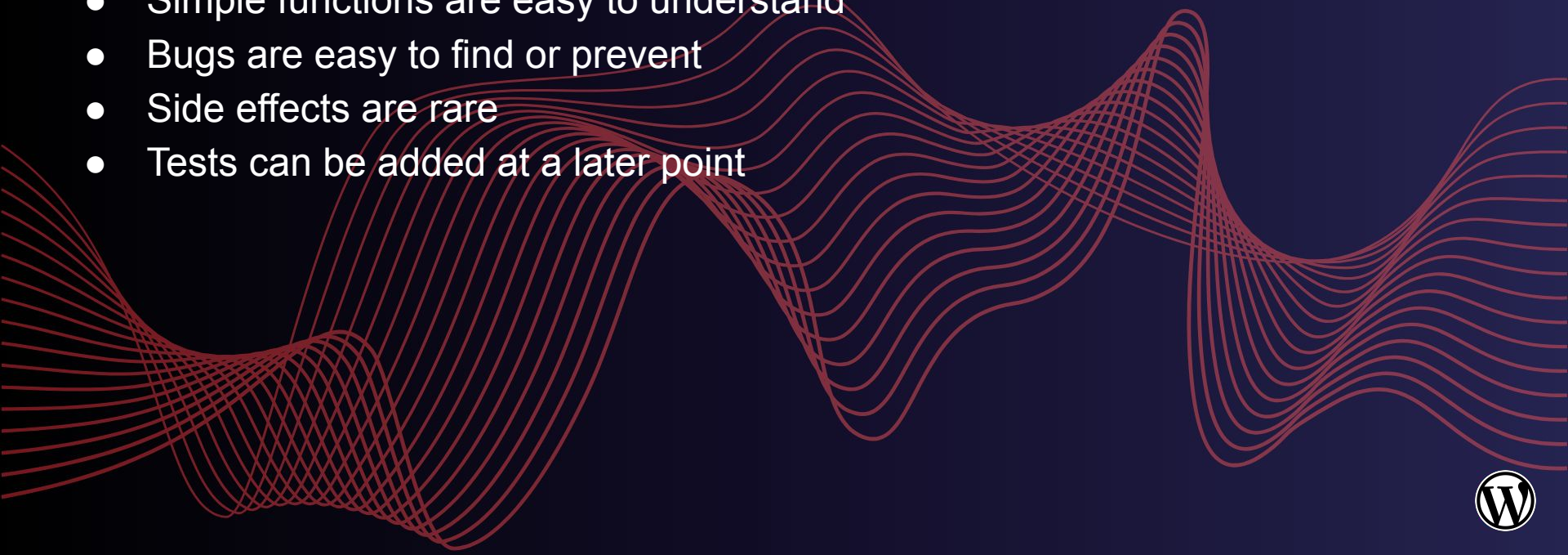
Q: Testable code



A: Testable code

Testable code or code following SOLID, is also good without test

- Simple functions are easy to understand
- Bugs are easy to find or prevent
- Side effects are rare
- Tests can be added at a later point



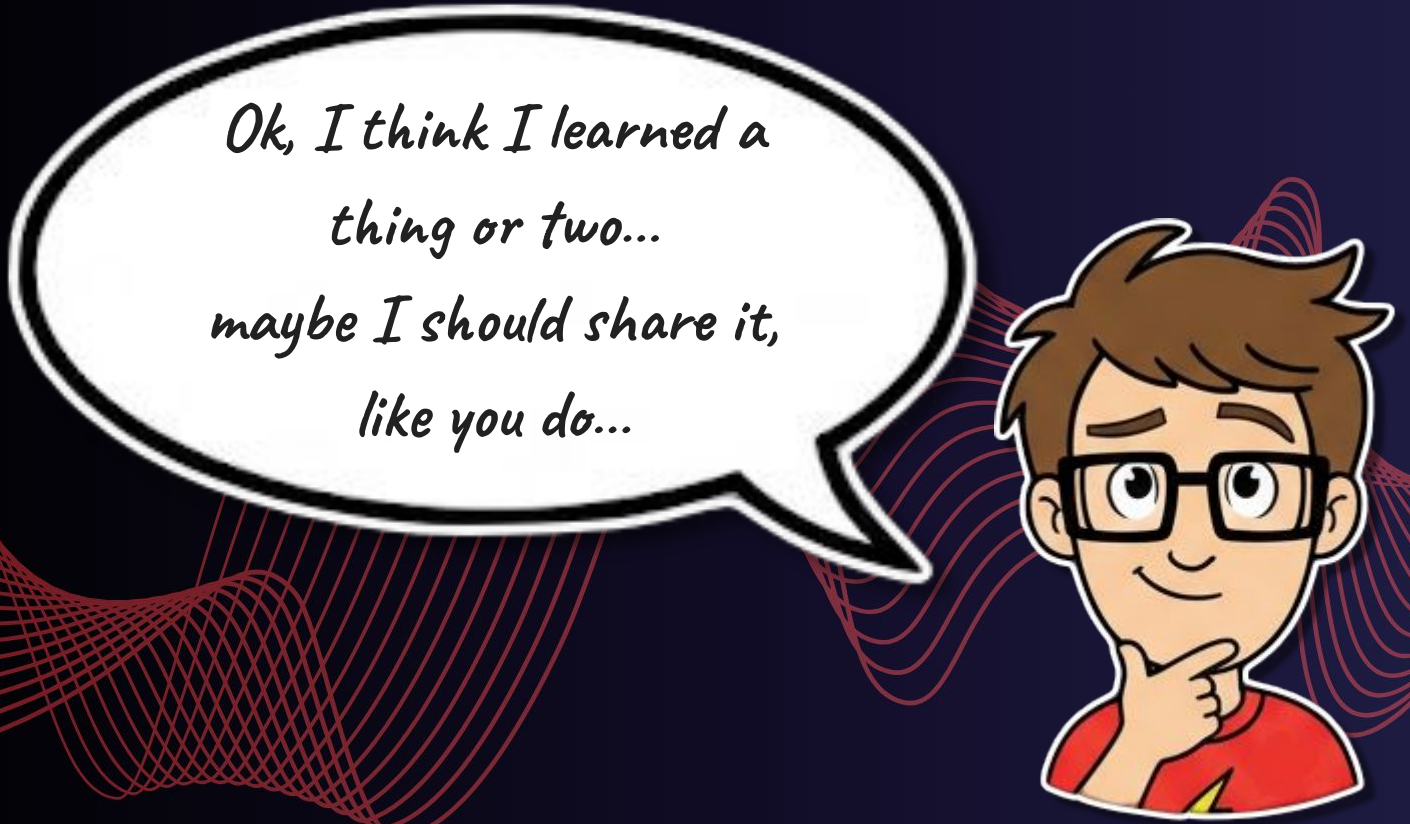
Q: A11y, I18n, SEO and more

A: A11y, I18n, SEO and more

In short words

But I’m out of time, so I can’t cover it today

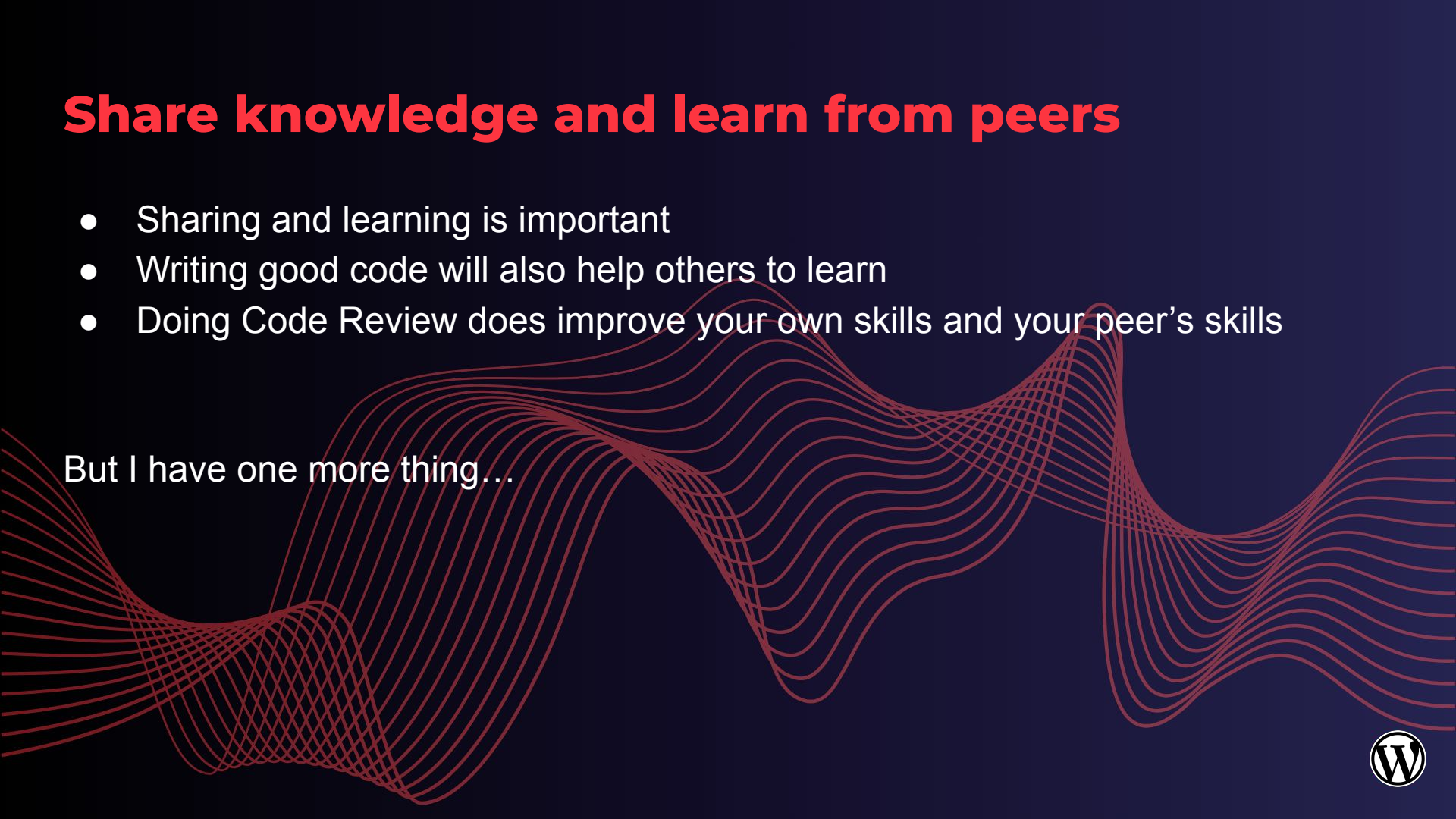
Share knowledge and learn from peers



Share knowledge and learn from peers

- Sharing and learning is important
- Writing good code will also help others to learn
- Doing Code Review does improve your own skills and your peer’s skills

But I have one more thing…



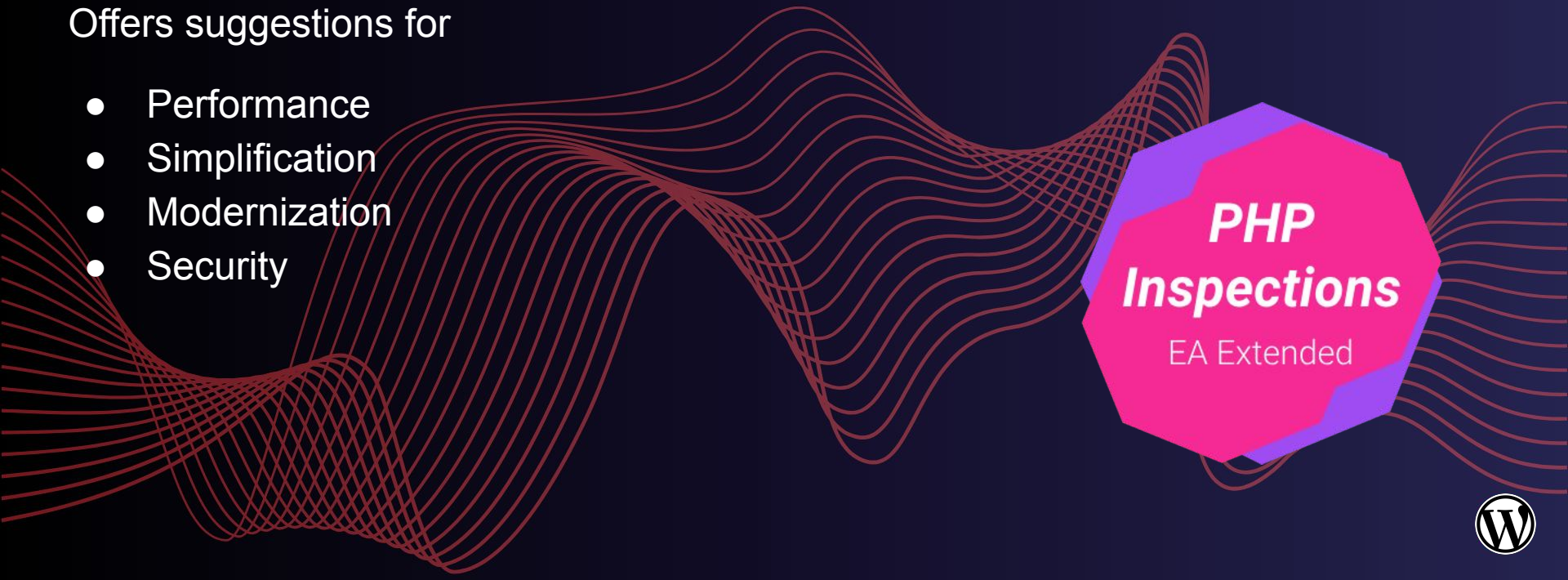
I am Sheldon... we all are.

Bonus Tip 1: PHP Inspections EA Extended

Static Code Analysis tool for PhpStorm (sorry VS Code People)

Offers suggestions for

- Performance
- Simplification
- Modernization
- Security



Bonus Tip 2: Composer & Autoloader

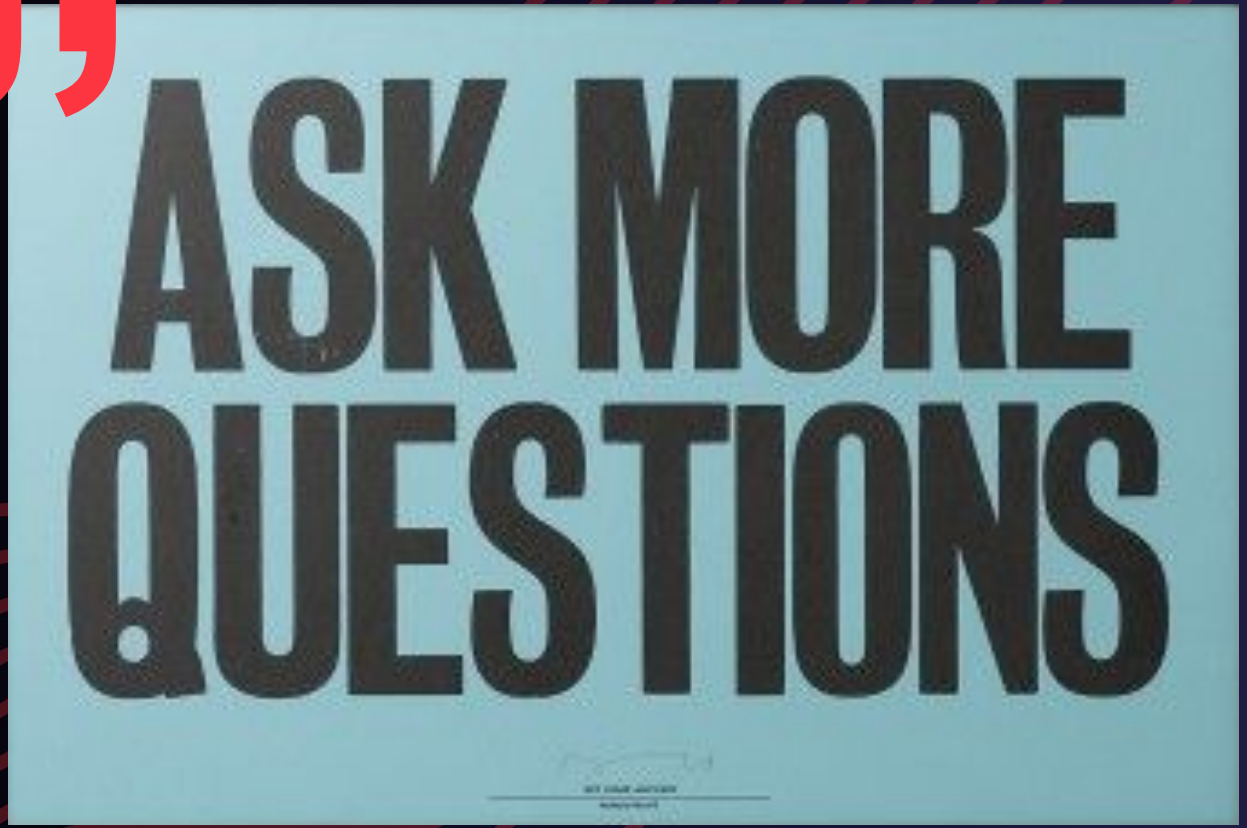
WordPress and Composer do work together

For themes and Plugins

- Autoloader
- Packages
- Coding Standards
- And much more

For Projects

- Bedrock https://roots.io/bedrock/



– Anthony Burill

Thank you for your attention

Slides at:
c13s.com/wordcamp

Further examples:
github.com/apermo

https://www.linkedin.com/in/christoph-daum/