drupalize me

What's New in Drupal 8

Object-Oriented PHP

Problems with Procedural Code

- To re-use code means to **duplicate code** into a new module, then make changes
- Code is **hard to test** when it's **tightly coupled**
- Function loading order is **magical and mysterious**
- StdClass Obj provides only values, **no hinting**

What is OO-PHP

- **Encapsulation** provides centralized access to methods and values
- **Inheritance** of values and function made possible
- **Interfaces** set expectations
- OO-PHP allows for **design patterns**
- Easier to **re-use, maintain, and extend**

Why use OO-PHP in Drupal 8?

- **Testable** code
- Better **organization**
- **Decoupling** reduces dependencies
- Interfaces & Data Types **improves insight**

How is OO-PHP done in D8?

- Much of the code base has been refactored to be **object-oriented**
- Each class is in **its own file**
- Classes must be **namespaced**
- **External libraries** introduced that Drupal extends
- **Design pattern**-based best practices in use

What makes OO-PHP exciting?

- Using **Inheritance**, I re-use code without duplication
- Inspecting **Interfaces**, I know what's expected
- By **Injecting Dependencies**, my code is decoupled and testable
- **Using best practices** makes me and my code happy

What do I need to know?

- **Language features of OO-PHP**, such as Constructors, Inheritance, Interfaces, Namespaces, and Type Hinting
- **Design patterns** and conventions such as **Dependency Injection, Services Container, PSR-4**