Coding Guidelines And Principles

目录

- PHP 编码规范
- JavaScript 编码规范
- 命名规范
- 编码原则
  - 保持简单直接（KISS）
  - 不要重复自己（DRY）
  - SOLID
  - 测试
    - 测试驱动开发（TDD）
  - 代码侦察原则
- 其他
  - 尝试/捕获“随机”功能
  - 压缩 .js 和 .css 文件
  - PHPDocs 与注释

PHP 编码规范

Yoast PHP 编码规范是一个 PHPCS 规则集，用于在所有 Yoast PHP 代码中强制执行 PHP 编码风格。

YoastCS 主要基于 WordPress 编码标准。在某些情况下，我们与 WPCS 存在差异。以下是两个示例：

1. 我们不强制使用尤达风格赋值。

```
// Yoast 风格
if ( $foo == 'bar' ) {
  echo 'It is Yoast style!';
```

```
}

// 尤达风格
if ( 'bar' == $foo ) {
  echo 'Yoda style is how this looks';
}
```

2. 我们将 `else` 放在新行。

```
// Yoast 风格:
if ( $foo == 'bar' ) {
  echo 'foo bar';
}
else {
  echo 'foo is not bar';
}

// WordPress 编码标准:
if ( $foo == 'bar' ) {
  echo 'foo bar';
} else {
  echo 'foo is not bar';
}
```

JavaScript 编码规范

我们的 JavaScript 编码规范 记录在 ESLint 规则集中。 JavaScript 与 PHP 的规则存在一些差异，例如我们使用驼峰命名法来命名类、函数、方法和变量。目前我们仅编写 ES6 代码，并通过 Webpack 和 Babel 等工具进行编译以确保向后兼容性。

命名规范

作为团队中的开发者，我们希望轻松理解代码的含义。为此，我们需要描述性的类名、函数名和变量名。以下是一些指导原则：

- 避免使用缩写，除非是像 `www`、`http`、`SEO` 等普遍使用的缩写。
- 不使用隐喻或晦涩的名称。唯一可接受的隐喻可能是 `needle` 和 `haystack`。

- 在命名对象和方法时，请记住对象是实体（名词），而方法是我们发送给它们的消息（命令式动词）。

编码原则

保持简单直接（KISS）

阅读代码时，应能轻松理解其逻辑。若陷入抽象、设计模式或其他编码标准的教条，你可能会为了符合标准而修改代码。这有点愚蠢，这也解释了 KISS 中最后一个 S 的含义。始终运用你的判断力，问问自己：这个改动是否让代码逻辑更易理解？如果不确定，随时可以寻求他人建议！

不要重复自己（DRY）

尽量避免在不同地方编写相同的功能。如果代码中存在错误，你不会希望修复多个地方的同一个错误。当出现非 DRY 代码时，很可能可以将其部分提取为可重用的函数或对象。仅在重复实际发生时进行 DRY 处理，并确保功能完全相同而非相似。否则，你可能会在之后花费宝贵的时间去解耦那些代码，而这恰恰是你最初试图节省的时间！

SOLID

SOLID 原则是一组指导代码应如何编写的原则。这些原则旨在帮助你以模块化的方式组织代码，使代码更易于理解，从而降低变更成本。SOLID 中的 S 代表单一职责原则，这可能是最容易理解的。一个类应该只有一个改变的理由，意味着一个类应该只负责一项工作。这些原则中的许多不仅适用于对象，也可以轻松地应用于函数和模块。更多内容请阅读链接文章。

测试

编写代码时，确保其按预期方式运行至关重要。为此，你应该编写测试。在 Yoast，我们使用 PHPUnit 进行 PHP 测试，使用 Jest 进行 JavaScript 测试。

关于测试的一些指导原则：

- 尽管通常只测试公共接口，但有时大量行为隐藏在私有接口中。在这种情况下，尝试通过测试替身等方式测试私有接口。
- 修复错误时，首先尝试在回归测试中重现该错误。回归测试是为了防止错误再次发生而编写的测试。之后只需让测试通过即可。

对于每个 PR，检查变更是否被单元测试覆盖。

测试驱动开发 (TDD)

TDD
是一种在编写实际实现代码之前先编写测试的原则。通过这种方式工作，你被迫提前思考要解决的问题。由于测试应该是独立且小巧的，你也已经在思考将要编写的实现功能的代码。虽然我们并不严格强制这种工作方式，但它是养成测试习惯的好方法。

代码侦察原则

重构应主要遵循代码侦察原则。这意味着你离开时留下的代码应比发现时更好。在修复错误或添加新功能时，
可以花些时间改进相关功能。当然，仅在确实需要时这样做，并记住保持简洁。

其他

尝试 /捕获 “随机 ”功能

某些功能具有随机的成功因素。例如：进行 API 调用时，你永远无法确定 API
是否可用。我们不希望软件因此中断。请将此类功能包装在 try/catch 块中。

压缩 .js 和 .css 文件
请确保你的 JS 和 CSS 文件已被压缩 /混淆，并在浏览器中加载压缩版本。我们的大多数仓库都配置了 Grunt
来帮你完成这项工作。只需运行 grunt build 即可！

PHPDocs 与注释

确保所有代码都使用 PHPDoc 进行文档化。必要时也使用行内注释，例如在描述复杂代码段时。

版本号 #2
由 壳壳虫 创建于 2026-03-26 17:08:40 UTC
由 壳壳虫 更新于 2026-03-26 17:08:40 UTC