□□ PHP Zero → Expert (with a light WordPress focus) — Table of Contents

> Structure: **Chapters** → **Subchapters** (you'll later request like `c2.3`).
> Each chapter builds forward, but key ideas are **revisited gently** so you don't have to constantly jump around. WordPress parallels are sprinkled throughout □□

2.4 Arrays (indexed, associative) and common operations
2.5 Control flow: `if/elseif/else`, `switch`, match expressions
2.6 Loops: `for`, `foreach`, `while`, `break/continue`
2.7 Practical mini-exercises (formatting, parsing, simple transforms)
2.8 WordPress parallel: arrays everywhere (hooks, query args, theme data)

---

3) Functions, Scope, and Working with Data □□

3.1 Defining functions, parameters, defaults, and return values
3.2 Scope, globals, static variables, and “why things disappear”
3.3 Passing by value vs by reference
3.4 Useful built-ins (strings, arrays, dates) without memorizing everything
3.5 Namespaces (why they matter even in small projects)
3.6 WordPress parallel: pluggable functions, naming, and avoiding collisions

---

4) HTTP, Forms, and Input Handling □□

4.1 HTTP essentials: methods, headers, status codes
4.2 Superglobals: `$_GET`, `$_POST`, `$_SERVER`, `$_COOKIE`, `$_FILES`
4.3 Forms end-to-end (build, submit, validate, respond)
4.4 Redirects and the PRG pattern (Post/Redirect/Get)
4.5 File uploads safely
4.6 WordPress parallel: admin forms, nonces, request handling conventions

---

5) Defensive PHP: Validation, Sanitization, and Security Basics □□

5.1 Threat model basics (what can go wrong)
5.2 Validation vs sanitization vs escaping (clear separation)

5.3 Output escaping for HTML (and why context matters)
5.4 Password hashing and authentication fundamentals
5.5 Sessions and cookies (secure defaults)
5.6 Common web vulnerabilities: XSS, CSRF, SQLi, SSRF (practical overview)
5.7 WordPress parallel: `sanitize_*`, `esc_*`, nonces, roles/capabilities

---

6) Working with Files, JSON, and Common Formats □□

6.1 Reading/writing files safely
6.2 Paths, directories, and portability
6.3 JSON encode/decode and pitfalls
6.4 CSV basics (import/export)
6.5 Date/time handling (timezone sanity)
6.6 WordPress parallel: media/filesystem API concepts, JSON in REST responses

---

7) Error Handling, Debugging, and Testing Mindset □□

7.1 Errors vs exceptions, and how PHP reports problems
7.2 Try/catch patterns and custom exceptions
7.3 Logging strategies (what to log, what not to log)
7.4 Debugging workflows (Xdebug basics, var_dump discipline)
7.5 Intro to automated testing concepts (unit vs integration)
7.6 WordPress parallel: WP_DEBUG, debug.log, common debugging workflows

---

8) Object-Oriented PHP: From Practical to Pro □□

8.1 Classes/objects, properties, methods
8.2 Constructors, visibility, and encapsulation

8.3 Inheritance vs composition (when to use which)
8.4 Interfaces and abstract classes
8.5 Traits (pros/cons)
8.6 Static usage (when it’s fine, when it’s a trap)
8.7 WordPress parallel: OOP patterns in plugins, service classes, admin pages

---

9) Modern PHP Practices: Autoloading, Composer, and Standards ⚙

9.1 Composer fundamentals (packages, versions, lockfiles)
9.2 Autoloading and PSR-4
9.3 Common PSRs (PSR-12, PSR-3, PSR-4) and why they help
9.4 Dependency injection (practical introduction)
9.5 Configuration patterns (env, config files)
9.6 WordPress parallel: Composer in plugins/themes, when to bundle dependencies

---

10) Databases with PHP: SQL, PDO, and Data Modeling □□

10.1 Relational database basics and schema thinking
10.2 SQL essentials (SELECT/INSERT/UPDATE/DELETE, joins)
10.3 PDO: prepared statements and safe queries
10.4 Transactions and consistency
10.5 Basic modeling: one-to-many, many-to-many
10.6 Performance basics: indexes and query shape
10.7 WordPress parallel: `$wpdb`, custom tables, and when *not* to create them

---

11) Building a Web App: Routing, Controllers, and Views □□

11.1 Simple routing (front controller pattern)
11.2 Organizing “MVC-ish” code without overengineering
11.3 Templating approaches (plain PHP templates done right)
11.4 Handling errors and 404s cleanly
11.5 Pagination, filters, and query parameters
11.6 WordPress parallel: templates, the loop conceptually, template hierarchy mindset

---

12) APIs: Consuming and Serving HTTP Services □□

12.1 Making HTTP requests (cURL / modern clients)
12.2 REST basics and JSON API conventions
12.3 Authentication patterns (API keys, OAuth overview)
12.4 Building a simple JSON API in PHP
12.5 Versioning and backwards compatibility
12.6 WordPress parallel: WP REST API usage and custom endpoints

---

13) Performance, Caching, and Scalability ⚡

13.1 Profiling mindset: find bottlenecks before “optimizing”
13.2 Opcode cache (OPcache) basics
13.3 Caching layers: in-memory, file-based, HTTP caching
13.4 Efficient I/O, streaming, and avoiding large memory spikes
13.5 Async-ish patterns (queues, cron) at a practical level
13.6 WordPress parallel: transients, object cache, page cache, cron behavior

---

14) WordPress Development Track (Sprinkled Knowledge → Structured Practice) □□

14.1 WordPress architecture overview (request lifecycle, hooks)
14.2 Theme fundamentals: templates, enqueueing, child themes
14.3 Plugin fundamentals: headers, structure, activation hooks
14.4 Hooks deep dive: actions vs filters, priorities, args
14.5 Security in WP: capabilities, nonces, escaping
14.6 Custom Post Types & Taxonomies (mental model + practice)
14.7 Meta fields and options (when to use what)
14.8 WP REST API: extend and consume
14.9 Data: `WP_Query`, `$wpdb`, and performance considerations
14.10 Practical patterns: settings pages, admin UI, shortcodes, blocks overview

---

15) Professional Practices: Architecture, Maintenance, and Delivery □□

15.1 Designing for change: boundaries, modules, and refactors
15.2 Documentation that stays useful (READMEs, docblocks)
15.3 Version control workflows (Git) for solo + teams
15.4 CI basics (linting, tests, static analysis)
15.5 Deployment overview (shared hosting, VPS, containers)
15.6 Observability basics: logs, metrics, error reporting
15.7 WordPress parallel: release discipline for plugins/themes and compatibility

---

16) Capstone Projects (Pick One or Do All) □□□

16.1 Pure PHP mini-app: form-heavy CRUD app with auth + admin
16.2 API project: PHP JSON API + a tiny client
16.3 WordPress plugin: production-style plugin (settings, CPT, REST endpoint)
16.4 WordPress theme: custom theme with performance + security best practices
16.5 Hardening & polish: testing, docs, deployment checklist

How to proceed

Send something like `c2.3` and I'll generate the full lesson for *Chapter 2, Subchapter 3* (including explanations, examples, a few exercises, and small WordPress parallels where relevant).

Revision #3
Created 2026-04-10 01:07:41 UTC by art10m
Updated 2026-04-10 01:42:06 UTC by art10m