WordPress Fundamentals

A Complete 10-Chapter Web Platforms Course

Topics covered:

Installation & setup · the dashboard & Block Editor · themes & plugins

Media, categories & tags, menus & widgets · users & roles · comments

Ongoing maintenance, updates & backups

Written for the site owner — no PHP required

Companion course: WordPress Intermediate/Advanced opens every black box this course deliberately left closed

Exercises: 30 hands-on scenarios with worked solutions

Format: A4 · Dark-theme code examples

Philip Osztromok · Generated with Claude

Table of Contents

CHAPTER 1 OF 10

What WordPress Actually Is

WordPress Fundamentals

Chapter 1 · What WordPress Actually Is

"WordPress" is one name attached to two genuinely different things, running on top of a stack this site has already covered in real depth from three separate directions. This chapter sorts out both — what you're actually about to learn, and what it's actually built on — before any dashboard tour begins.

WordPress.org vs. WordPress.com — Two Very Different Things Behind One Name

WordPress.org is free, open-source software — a program you install on your own web hosting, that you fully own and control. **WordPress.com** is a commercial hosted service, run by a company called Automattic, built *using* that same open-source software but wrapped in its own hosting, its own rules, and its own tiered pricing. Same underlying name, same underlying code — genuinely different products.

WORDPRESS.ORG (SELF-HOSTED)	WORDPRESS.COM (HOSTED SERVICE)
You provide your own hosting and install the software yourself	Automattic hosts it for you — no server of your own required
Any plugin or theme can be installed, with no restrictions	Custom plugins/themes are restricted or blocked entirely on lower-tier plans
Full control over updates, backups, and configuration	Much of this is handled — and locked down — by the platform
Free to download; you pay only for hosting	Free tier exists, but real capability requires a paid plan

THIS COURSE IS ENTIRELY ABOUT WORDPRESS.ORG

Self-hosted WordPress is the version with real power and real flexibility — the one worth understanding deeply as a site owner or a future developer. Every chapter in this course, and in `WordPress Intermediate/Advanced`, assumes the self-hosted, WordPress.org version unless stated otherwise.

Why WordPress's Market Share Actually Matters

A GENUINELY NOTABLE, REAL STATISTIC

> WordPress powers a substantial share of the entire web — commonly cited figures put it at well over 40% of all websites, and a clear majority of sites that run on any identifiable content management system at all. This isn't a niche tool; it's one of the most widely deployed pieces of software on the internet.

That scale has real, practical consequences worth knowing before you invest time learning it:

- **An enormous plugin and theme ecosystem** — tens of thousands of free and paid options covering almost any feature a site might need
- **A huge base of documentation and community support** — a problem you hit has very likely already been solved and written up publicly
- **A genuinely marketable, employable skill** — WordPress site management and development are real, in-demand job categories, not a purely academic exercise

The Stack Underneath — LAMP, and Three Courses Already on This Site

WordPress isn't a mysterious black box — it's a PHP application, storing its content in a MySQL (or MariaDB) database, typically served by Apache or Nginx on Linux. That's the classic **LAMP** stack (Linux, Apache, MySQL, PHP), and this site has already covered every layer of it, separately, in real depth.

LAMP LAYER	WHAT WORDPRESS USES IT FOR	ALREADY COVERED BY
Linux + Apache	The server WordPress's files live on and get served from	*Setting Up a Web Server on Debian*
MySQL	Every post, page, comment, user, and setting is a row in a MySQL database	*MySQL Installation & Administration*, *MySQL Foundations*
PHP	WordPress core, every theme, and every plugin is PHP code, executed on each page request	*PHP Fundamentals*

> **YOU DON'T NEED ANY OF THAT TO START THIS COURSE**
>
> *WordPress Fundamentals* is deliberately usable as a pure site-owner's course — managing content, themes, and plugins through the dashboard, with no PHP, no SQL, and no server administration required at all. The LAMP-stack knowledge above becomes directly relevant only once *WordPress Intermediate/Advanced* opens the hood on theme and plugin development — this chapter names the connection early so it's there when you need it, not because you need it yet.

What This Course Covers

Installation and initial setup, the dashboard and its core concepts, the Block Editor, themes, plugins, media and content organization, user roles, comments, and ongoing maintenance — everything a site owner

genuinely needs, in the order it's actually needed. *WordPress Intermediate/Advanced* picks up from there with real PHP development: custom themes, custom post types, plugin development, and security hardening.

Hands-On Exercises

EXERCISE 1

A friend says they're "using WordPress" for their blog and mentions they can't install any custom plugins. Explain what this tells you about which version of WordPress they're actually using, and why.

View solution

EXERCISE 2

Explain, in your own words, what each letter of LAMP corresponds to in a running WordPress site, and name one specific thing WordPress stores in the "M" layer.

View solution

EXERCISE 3

Explain why this chapter argues that WordPress's large market share is practically useful to know about, beyond simply being an interesting statistic.

View solution

CHAPTER 1 QUICK REFERENCE

- **WordPress.org** — free, self-hosted software, full control; this course's entire focus
- **WordPress.com** — a hosted commercial service built on the same software, with real restrictions on lower tiers
- WordPress powers well over 40% of all websites — a huge ecosystem, huge community support, and a real, marketable skill
- WordPress runs on the **LAMP** stack — Linux/Apache, MySQL, PHP — already covered separately by this site's own server, database, and PHP courses
- No PHP/SQL/server knowledge is required to start this course — that becomes relevant in *WordPress Intermediate/Advanced*
- **Next chapter:** Installation & Initial Setup

CHAPTER 2 OF 10

Installation & Initial Setup

WordPress Fundamentals

Chapter 2 · Installation & Initial Setup

WordPress Fundamentals 1 named the LAMP stack WordPress runs on without asking you to build any of it yourself. This chapter is the one moment in this course where that stack becomes directly visible — not because you need to build it, but because knowing what's happening underneath makes every setting in the next few chapters make sense rather than feel arbitrary.

Hosting Requirements

Any host advertising WordPress support needs to provide, at minimum: a current PHP version, a MySQL or MariaDB database, and enough disk space for WordPress core plus whatever media and plugins accumulate over time. HTTPS support is effectively mandatory today, not optional — covered in full technical depth in *HTTPS/TLS Fundamentals*, and specifically for a Debian-hosted setup, in *Setting Up a Web Server on Debian*'s own Let's Encrypt chapter.

HOSTING TYPE	WHAT YOU GET
Shared hosting	Cheapest, easiest to start with; resources shared with other sites on the same server
Managed WordPress hosting	A host specifically tuned for WordPress — automatic updates, built-in caching, staging environments
A VPS you administer yourself	Full control, the most work — effectively building the exact stack *Setting Up a Web Server on Debian* covers from scratch

One-Click Installers — The Common Path

Most hosting providers offer a one-click WordPress installer (Softaculous is a common example) built into their own control panel. It handles creating the MySQL database, downloading WordPress, and running the setup wizard automatically — for the overwhelming majority of site owners, this is genuinely the right choice, not a shortcut to feel guilty about.

The Manual Install — What Actually Happens Underneath

Understanding the manual process is worth doing once, even if you'll use a one-click installer afterward — it demystifies exactly what that installer is automating for you.

1. Create a MySQL database and a database user with privileges on it — the same fundamental operation covered in *MySQL Installation & Administration*
2. Download the WordPress software and upload its files to your hosting
3. Visit the site in a browser — WordPress detects there's no configuration yet and launches its own famous "five-minute install" wizard
4. Enter the database connection details when prompted, then choose a site title, an admin username, and an admin password

> **"FIVE-MINUTE INSTALL" ISN'T MARKETING EXAGGERATION**
>
> This nickname is decades old at this point and still broadly accurate — the actual installation wizard genuinely is that fast. Almost everything covered in the rest of this course happens after this point, inside the dashboard, not during installation itself.

wp-config.php At a Glance

This single file, generated during installation, is WordPress's own core configuration — the first file WordPress reads on every single page load, before anything else happens.

```
define( 'DB_NAME', 'wordpress_db' );
define( 'DB_USER', 'wp_user' );
define( 'DB_PASSWORD', 'a-strong-password' );
define( 'DB_HOST', 'localhost' );

$table_prefix = 'wp_';

define( 'WP_DEBUG', false );
```

- **DB_NAME / DB_USER / DB_PASSWORD / DB_HOST** — the database connection details, the exact information a manual install prompts for
- **Authentication unique keys and salts** — long random strings (generated automatically, not shown above) that make WordPress's own login cookies harder to forge
- **$table_prefix** — every WordPress database table starts with this prefix (`wp_` by default); changing it away from the default is a small, genuine hardening step, revisited properly in *WordPress Intermediate/Advanced*'s own security chapter
- **WP_DEBUG** — toggles PHP error/warning output; left `true` accidentally on a live site can leak technical details to visitors, so this should always be `false` once a site is genuinely live

An Initial Security Checklist

A FEW MINUTES NOW, DONE ONCE

- **A strong, unique admin password** — this is the single highest-value step, and WordPress's own install wizard will suggest one
- **A non-obvious admin username** — never literally `admin`, which is the first guess in any automated attack; the real reasoning behind this is covered fully once WordPress Fundamentals 8 introduces the role hierarchy
- **HTTPS enabled from day one** — per *HTTPS/TLS Fundamentals*, this protects login credentials and all site traffic in transit, not just the checkout page of an online store
- **A non-default table prefix**, if installing manually — a small, genuine layer of obscurity against automated attacks that target the default `wp_` prefix specifically

This is the beginner-facing version of a much deeper conversation — real, code-level security (nonce verification, capability checks, output escaping) is covered in full in *WordPress Intermediate/Advanced*'s own dedicated Security Hardening chapter.

Hands-On Exercises

EXERCISE 1

Explain what a one-click installer is actually doing on your behalf, using the manual-install steps from this chapter as your reference.

View solution

EXERCISE 2

A site has been live for months and a developer notices WP_DEBUG is still set to true. Explain why this is a real problem, not just a stylistic oversight.

View solution

EXERCISE 3

Explain why an admin username of "admin" is considered a security weakness, connecting your answer to what an automated attack is actually doing.

View solution

CHAPTER 2 QUICK REFERENCE

- Hosting needs: current PHP, MySQL/MariaDB, disk space, and HTTPS — effectively mandatory today
- One-click installers automate database creation, file upload, and the setup wizard — the right choice for most site owners
- Manual install: create the database → upload files → run the five-minute install wizard
- **wp-config.php** — the core config file: database credentials, auth keys/salts, table prefix, WP_DEBUG
- Initial security: a strong admin password, a non-"admin" username, HTTPS from day one, a non-default table prefix
- **Next chapter:** The Dashboard & Core Concepts

CHAPTER 3 OF 10

The Dashboard & Core Concepts

WordPress Fundamentals

Chapter 3 · The Dashboard & Core Concepts

With WordPress installed, this chapter covers the two things every later chapter assumes you already know: how to find your way around the admin dashboard, and the single most important content-model distinction in all of WordPress — Posts versus Pages.

A Tour of the Admin Dashboard

Logging in lands you on `wp-admin` — the dashboard's home screen, with a left-hand sidebar that stays consistent across every screen in WordPress:

SIDEBAR ITEM	WHAT IT'S FOR
Posts	Chronological content — covered in depth below
Media	Every uploaded image, video, and document — covered fully in WordPress Fundamentals 7
Pages	Static, standalone content — also covered below
Comments	Reader comments awaiting moderation — covered in WordPress Fundamentals 9
Appearance	Themes, menus, widgets — covered in WordPress Fundamentals 5 and 7
Plugins	Installed and available plugins — covered in WordPress Fundamentals 6
Users	Accounts and their roles — covered in WordPress Fundamentals 8
Settings	Site-wide configuration — general settings, permalinks, discussion rules

The dashboard home screen itself shows an "At a Glance" summary (post/page/comment counts), recent activity, and a Quick Draft box for jotting down a post idea without leaving the home screen.

Posts vs. Pages — The Distinction Everything Else Depends On

Posts are chronological content — blog entries, news updates, anything meant to be read in a reverse-chronological stream and organized by category or tag. They're what feeds an RSS feed and what a typical "blog" listing page shows.

Pages are static, standalone content with no inherent date-sensitivity — an About page, a Contact page, a Privacy Policy. Pages can be organized hierarchically (a parent page with child pages beneath it) and have no categories or tags by default.

POSTS	PAGES
Chronological, date-driven	Static, not date-driven
Organized via categories and tags	Organized hierarchically (parent/child)
Included in the site's RSS feed	Not included in the RSS feed
Example: a blog entry, a news update	Example: About, Contact, Privacy Policy

A TECHNICAL DETAIL WORTH KNOWING NOW

Posts and Pages aren't stored in two separate systems — they live in the exact same underlying database table, distinguished by a single column that records which "post type" each row is. WordPress Intermediate/Advanced 4 shows that this same mechanism is exactly what lets a developer define an entirely new, custom content category beyond just these two — a portfolio item, a product, a testimonial — using the identical underlying system.

Which One Should You Use?

A QUICK DECISION GUIDE

Ask: is this content tied to a specific point in time, meant to appear in a chronological stream? Use a **Post**. Is it standalone, timeless, and meant to always be reachable the same way (like a fixed page in a site's main navigation)? Use a **Page**. Getting this wrong doesn't break anything technically, but it does mean content shows up in the wrong place — a Contact page accidentally published as a Post would show up in the blog feed alongside articles, which is rarely what anyone actually wants.

Post & Page Statuses

Both Posts and Pages share the same set of statuses:

- **Draft** — saved but not visible to visitors
- **Pending Review** — submitted by a contributor, awaiting approval (relevant once WordPress Fundamentals 8 covers roles)
- **Scheduled** — set to publish automatically at a future date/time
- **Published** — live and publicly visible
- **Private** — published, but visible only to logged-in users with sufficient permission

Hands-On Exercises

EXERCISE 1

A site owner publishes their "Contact Us" information as a Post instead of a Page. Explain what would go visibly wrong as a result, using this chapter's own material.

View solution

EXERCISE 2

Explain what this chapter means by saying Posts and Pages are "the same underlying system," and why that detail matters for understanding what a custom post type will turn out to be.

View solution

EXERCISE 3

Explain the difference between a "Scheduled" and a "Private" post status, and describe a realistic situation where each one would be the right choice.

View solution

CHAPTER 3 QUICK REFERENCE

- The dashboard sidebar (Posts, Media, Pages, Comments, Appearance, Plugins, Users, Settings) stays consistent across every screen
- **Posts** — chronological, categorized/tagged, feed into RSS (blog entries, news)
- **Pages** — static, hierarchical, no categories/tags by default (About, Contact)
- Posts and Pages share the same underlying database table — a detail that directly explains custom post types later
- Shared statuses: Draft, Pending Review, Scheduled, Published, Private
- **Next chapter:** The Block Editor (Gutenberg), In Depth

CHAPTER 4 OF 10

The Block Editor (Gutenberg), In Depth

WordPress Fundamentals

Chapter 4 · The Block Editor (Gutenberg), In Depth

Every Post and Page from WordPress Fundamentals 3 is actually written using this chapter's own subject: the Block Editor, nicknamed "Gutenberg" after the inventor of the printing press. This chapter goes past "click here to add text" into how the editor is actually structured, and two features — patterns and reusable blocks — that look similar on the surface but work in genuinely different ways.

What Gutenberg Actually Is

The Block Editor became WordPress's own default editor in WordPress 5.0 (2018), replacing the older "Classic Editor" — a single large text box closer to a traditional word processor.

> **AN HONEST NOTE ON A GENUINELY CONTENTIOUS CHANGE**
>
> This transition was, and to some degree still is, controversial among long-time WordPress users — many found the shift from a familiar single text box to a block-based structure disruptive, and the official Classic Editor plugin (restoring the old behavior) remained in heavy use for years afterward. This course teaches the Block Editor because it's the current default and the direction WordPress itself is actively developing, not because the older approach was without genuine merit.

Blocks — The Fundamental Unit

Everything in the content area is a **block** — a paragraph, a heading, an image, a list, a quote, a button, a columns layout. Each block carries its own settings and toolbar, and blocks can be nested inside one another (a Columns block, for example, contains multiple Column blocks, each of which can hold any other block inside it).

- **Inserting a block** — the "+" inserter, or typing `/` followed by a block name to search the block library directly from the keyboard
- **Rearranging blocks** — drag handles, or the up/down arrows in each block's own toolbar
- **Block-specific settings** — the right-hand sidebar shows settings specific to whichever block is currently selected, separate from the "Document" tab covering post-level settings (visibility, publish date, categories, featured image)

Patterns — Ready-Made Block Arrangements

A **pattern** is a pre-designed group of blocks — a "hero section" combining a heading, paragraph, and button in a specific layout, for example — inserted as a single starting point rather than assembled block by block from scratch.

Reusable Blocks — A Genuinely Different Thing, Despite Looking Similar

A **reusable block** (in newer WordPress versions, sometimes labeled a "synced pattern") is also a saved group of blocks — but with one critical difference from an ordinary pattern: editing a reusable block updates it *everywhere it's used, simultaneously*. A pattern, once inserted, becomes an ordinary, independent set of blocks you can freely edit without affecting anything else; a reusable block stays permanently linked across every page it appears on.

PATTERN	REUSABLE BLOCK (SYNCED PATTERN)
A one-time starting template	A permanently linked, shared piece of content
Editing one instance affects only that instance	Editing one instance updates every instance, everywhere, at once
Best for: a layout you want to reuse but customize differently each time	Best for: content that must always stay identical everywhere — a site-wide notice, a shared call-to-action

A CONCRETE EXAMPLE OF THE DIFFERENCE

A "Book Review" pattern, containing a rating block and a summary paragraph, is inserted on ten different review posts — editing one review's rating has no effect on the other nine, since each is now its own independent set of blocks. A "Newsletter Signup" reusable block, inserted at the bottom of every blog post, updates its own text and button on every single post at once the moment it's edited anywhere — exactly the intended behavior when the goal is one consistent, site-wide element rather than ten independently editable copies.

WHERE THIS BECOMES A GENUINE GOTCHA

Someone who doesn't realize a block is a reusable/synced one can make what they think is a small, one-off edit to a single page — and unintentionally change that same content everywhere else it's used. Before editing an unfamiliar block that behaves unexpectedly like it's "linked" to something else, it's worth checking whether it's actually a reusable block rather than an ordinary one.

A First Look at Full Site Editing

Modern WordPress themes built specifically for it — "block themes" — extend the Block Editor beyond just Post and Page content into the site's own header, footer, and overall template structure, a capability known as

Full Site Editing (FSE). This is a genuinely newer, still-evolving part of WordPress, and many widely-used themes are still "classic" themes where FSE doesn't fully apply.

> **SCOPE NOTE**
>
> This course covers FSE only at the level of recognizing it and knowing it exists. Building a theme from scratch — whether via FSE or the traditional PHP template approach — is covered properly in WordPress Intermediate/Advanced, starting with its own opening chapter on theme anatomy.

Hands-On Exercises

EXERCISE 1

A site owner inserts the same "Contact Us" reusable block at the bottom of 15 different pages, then edits the phone number on one of them expecting only that page to change. Explain what actually happens, and why.

View solution

EXERCISE 2

Explain, using this chapter's own material, when a pattern is the better choice over a reusable block, and give a realistic example distinct from the ones already in this chapter.

View solution

EXERCISE 3

Explain why this chapter includes an honest note about the Classic Editor transition rather than simply presenting the Block Editor as an uncontroversial improvement.

View solution

CHAPTER 4 QUICK REFERENCE

- The Block Editor ("Gutenberg") has been WordPress's default since version 5.0 (2018), replacing the single-textarea Classic Editor
- Every piece of content is a **block** — insert via "+" or `/`, blocks can nest inside one another
- **Patterns** — a one-time starting template; editing an inserted instance affects only that instance
- **Reusable blocks (synced patterns)** — permanently linked; editing any instance updates every instance everywhere at once

- Confusing the two is a real, common gotcha — check before editing an unfamiliar block that behaves unexpectedly
- **Full Site Editing** — extends blocks to headers/footers/templates on modern block themes; full coverage deferred to WordPress Intermediate/Advanced
- **Next chapter:** Themes

CHAPTER 5 OF 10

Themes

WordPress Fundamentals

Chapter 5 · Themes

Everything written so far — Posts, Pages, blocks — has a home to live in visually, and that home is a theme. This chapter treats a theme deliberately as a black box: how to choose, install, and customize one, without yet opening it up to see how it actually works. That part is WordPress Intermediate/Advanced's own job, starting with its very first chapter.

What a Theme Actually Is (For Now, a Black Box)

A theme is a package of files controlling a site's visual appearance and layout — colors, typography, page structure, how a blog listing or a single post is arranged on screen. Crucially, a theme doesn't hold your content. Every Post and Page, per WordPress Fundamentals 3's own material, lives in the MySQL database, entirely separate from whichever theme happens to be active. Switching themes changes how that content looks, never what it actually is.

Choosing a Theme

TYPE	WHAT TO KNOW
Free themes	Available directly from the official WordPress.org theme directory, searchable right inside the dashboard
Premium/paid themes	Sold through third-party marketplaces, often with more built-in design options and dedicated support
Block themes	Built specifically for the Full Site Editing capability previewed in WordPress Fundamentals 4 — headers, footers, and full page templates are all editable as blocks
Classic themes	Still very widely used — page structure is defined in PHP template files rather than editable blocks (the exact subject of WordPress Intermediate/Advanced's own opening chapters)

BEFORE INSTALLING ANY THEME, CHECK

When it was last updated (an abandoned theme is a real security and compatibility risk, previewing WordPress Fundamentals 10's own maintenance material), its rating and number of active installs, and whether it's compatible with your current WordPress version.

Installing a Theme

From the dashboard: **Appearance → Themes → Add New**. From there you can search the official directory directly, or upload a theme's `.zip` file if it came from a premium marketplace. Installing a theme doesn't activate it — a separate "Activate" step actually switches the live site over to it, and only one theme can be active at a time.

Customizing a Theme

Appearance → Customize opens the Customizer — a live-preview editor for classic themes, showing changes on the actual site as you make them: site identity (logo, title, tagline), color schemes, menu assignment, and homepage settings (a static page vs. a stream of latest posts, directly connected to WordPress Fundamentals 3's own Posts-vs-Pages material).

BLOCK THEMES REPLACE THIS WITH THE SITE EDITOR

On a block theme, much of what the Customizer used to handle moves into the Site Editor instead — the same Full Site Editing capability WordPress Fundamentals 4 previewed, now applied to the whole site's structure rather than just individual post/page content. Which interface you see depends entirely on which type of theme is currently active.

Child Themes — Why They Matter (A Concept, Not Yet a How-To)

A **child theme** inherits all the styling and functionality of a "parent" theme, while letting you make your own customizations safely, in a separate place.

THE REAL PROBLEM THIS SOLVES

Directly editing a theme's own files is a genuine, common WordPress mistake: the next time that theme receives an official update, every one of those direct edits gets silently overwritten and lost. A child theme keeps customizations in a completely separate set of files that an update to the parent theme never touches — the safe way to customize anything beyond what the Customizer or Site Editor already exposes.

DELIBERATELY LEFT AS A CONCEPT FOR NOW

This chapter stops at understanding why child themes exist. Actually building one — the real file structure, a `style.css` header declaring the parent theme, and the PHP needed to properly load the parent's own styles — is covered directly in WordPress Intermediate/Advanced's own opening chapter, which formally opens the black box this chapter deliberately left closed.

Hands-On Exercises

EXERCISE 1

A site owner switches their active theme from one design to a completely different one. Explain what happens to their existing blog posts, using this chapter's own material.

View solution

EXERCISE 2

A developer directly edits a theme's own PHP files to add a custom feature. Weeks later, the theme author releases an update, and the custom feature disappears entirely. Explain exactly what happened and what should have been done instead.

View solution

EXERCISE 3

Explain why a site's active theme being a "block theme" or a "classic theme" changes which customization interface a site owner actually sees.

View solution

CHAPTER 5 QUICK REFERENCE

- A theme controls appearance only — content lives separately in the database and survives any theme switch
- Free (WordPress.org directory) vs. premium (marketplaces); block themes (Full Site Editing) vs. classic themes (PHP templates)
- Check last-updated date, ratings, and active installs before installing any theme
- Install via Appearance → Themes → Add New; installing ≠ activating
- The **Customizer** (classic themes) or **Site Editor** (block themes) handle live customization
- **Child themes** — safely customize without losing changes on the next parent-theme update; the how-to is deferred to WordPress Intermediate/Advanced's own opening chapter
- **Next chapter:** Plugins

CHAPTER 6 OF 10

Plugins

WordPress Fundamentals

Chapter 6 · Plugins

Themes handled appearance in WordPress Fundamentals 5. This chapter covers the other half of "extending WordPress without touching its core files" — plugins, which add or change functionality rather than looks. Like themes, plugins are treated here as a trusted black box you install and use; WordPress Intermediate/Advanced's own plugin-development chapter is the one that builds one from scratch.

What a Plugin Actually Is

A plugin is PHP code that extends or modifies what WordPress can do — a contact form, an SEO toolkit, a security scanner — without ever touching WordPress's own core files directly. Plugins do this by hooking into specific points in WordPress's own execution, through a mechanism called actions and filters. This chapter treats that mechanism as a black box for now; WordPress Intermediate/Advanced 5 opens it up in full, framing it as the single most important WordPress-specific concept for a developer — and explicitly building, from scratch, the kind of tool this chapter only ever installs as a finished product.

Finding & Vetting a Plugin

Like themes, plugins come from the official WordPress.org directory (searchable directly in the dashboard) or from premium marketplaces. Unlike a theme, though, a bad plugin choice can genuinely break site functionality or open a real security hole — vetting matters more here, not less.

CHECK	WHY IT MATTERS
Last updated date	An abandoned plugin is a real compatibility and security risk — the same concern flagged for themes in WordPress Fundamentals 5
"Tested up to" version	Whether the plugin author has confirmed compatibility with your current WordPress version
Active installs & ratings	A rough proxy for how battle-tested a plugin actually is in the real world
Open support threads	How many unresolved issues exist, and how responsive the developer actually is to them

Installing, Activating — and a Real Difference From Themes

From the dashboard: **Plugins** → **Add New** to search the directory, or upload a `.zip` file directly. As with themes, installing and activating are two separate steps.

> **THE GENUINE DIFFERENCE FROM THEME ACTIVATION**
>
> Only one theme can be active at a time — but any number of plugins can be active simultaneously. A typical real site runs somewhere between five and twenty active plugins at once, each contributing its own piece of functionality independently.

Plugin Conflicts — A Real, Common Problem

Because multiple plugins run simultaneously, two plugins occasionally modify the same thing in incompatible ways — producing a broken page, a fatal error, or unexpected behavior with no obvious cause.

> **THE STANDARD TROUBLESHOOTING TECHNIQUE**
>
> Deactivate every plugin, then reactivate them one at a time, checking the site after each one, until the problem reappears — the plugin just reactivated is the culprit. This is slow but genuinely reliable, and it's the first real diagnostic step any experienced WordPress user reaches for.

Essential Plugin Categories

CATEGORY	WHAT IT DOES
SEO (e.g. Yoast SEO, Rank Math)	Meta descriptions, XML sitemaps, readability checks — the practical, in-dashboard tooling for everything *SEO Fundamentals* covers conceptually
Caching (e.g. WP Super Cache, W3 Total Cache)	Speeds up page delivery by serving pre-built pages instead of regenerating them on every request — previewed here, covered properly against real performance metrics in WordPress Intermediate/Advanced 9
Forms (e.g. Contact Form 7, WPForms)	Contact forms, surveys, and submission handling without writing any PHP
Security (e.g. Wordfence, Sucuri)	Firewalling, malware scanning, and login-attempt limiting — the plugin-level version of what WordPress Intermediate/Advanced 8's own code-level security chapter covers directly

An Honest Caveat: Plugin Bloat

> **MORE PLUGINS ISN'T FREE**

Every active plugin adds real PHP code that runs on every relevant page load, and every plugin is also additional code that could contain a vulnerability. Installing plugins for every conceivable feature — rather than only the ones a site genuinely needs — has a real, measurable performance cost (directly relevant to *Performance & Core Web Vitals*'s own material) and a real, cumulative security cost (an unmaintained plugin is exactly the kind of risk named in *OWASP Top 10*'s own Vulnerable & Outdated Components category). Fewer, well-vetted plugins consistently beats many loosely-chosen ones.

Hands-On Exercises

EXERCISE 1

A site suddenly shows a fatal error after a new plugin is installed and activated. Describe, step by step, the troubleshooting approach this chapter recommends to find the actual cause.

View solution

EXERCISE 2

Explain why a site owner can have many plugins active at once but only one theme, using this chapter's own explanation of what each one actually does.

View solution

EXERCISE 3

A site owner installs 40 plugins "just in case they're useful someday." Explain the two distinct real costs this chapter identifies with that approach.

View solution

CHAPTER 6 QUICK REFERENCE

- Plugins extend functionality via actions/filters, without touching WordPress core — the mechanism itself is deferred to WordPress Intermediate/Advanced 5
- Vet before installing: last updated, "Tested up to," active installs/ratings, open support threads
- Unlike themes (one active at a time), any number of plugins can be active simultaneously
- Plugin conflicts: deactivate all, reactivate one at a time to isolate the culprit
- Essential categories: SEO, caching, forms, security
- Plugin bloat has real performance and security costs — fewer, well-vetted plugins beats many loosely-chosen ones

- **Next chapter:** Media & Content Organization

CHAPTER 7 OF 10

Media & Content Organization

WordPress Fundamentals

Chapter 7 · Media & Content Organization

Posts and Pages hold your content; themes and plugins shape what surrounds it. This chapter covers what ties everything together — the media you upload, how posts get organized for browsing, and the menus and small content blocks that make a site actually navigable.

The Media Library

Every image, video, and document uploaded to a site lands in the **Media Library** (Media → Library), a central place to browse, search, and reuse anything already uploaded rather than uploading duplicates.

- **Alt text** — a written description attached to each image, read aloud by screen readers and shown if an image fails to load; genuinely important for accessibility, not an optional metadata field
- **Automatically generated image sizes** — WordPress creates several resized versions of every uploaded image (commonly thumbnail, medium, large, and the original full size) so the right size can be used in different contexts without manual resizing
- **Featured image** — one image chosen to represent a Post, used in blog listings, social-media link previews, and anywhere a theme displays a post's own thumbnail

> **ALT TEXT ISN'T BUSYWORK**
>
> A visually impaired visitor using a screen reader relies entirely on alt text to know what an image actually shows — skipping it isn't a minor shortcut, it's the difference between an image being genuinely accessible and effectively invisible to part of your audience.

Categories vs. Tags — A Distinction Worth Getting Right

Both organize Posts specifically — per WordPress Fundamentals 3's own compare table, Pages don't use either by default — but they solve genuinely different organizational problems.

CATEGORIES	TAGS
Broad, structural — the site's main topic areas	Granular, cross-cutting — specific topics a post touches on

CATEGORIES	TAGS
Hierarchical (can have parent/child categories)	Flat — no hierarchy at all
Every post has at least one — falls back to "Uncategorized" if none is chosen	Entirely optional
Think: a book's table of contents	Think: a book's index

A CONCRETE EXAMPLE

A cooking blog might use categories for its top-level structure — `Breakfast`, `Dinner`, `Desserts` — each post belonging to exactly one or two. Tags would then cross-cut those categories entirely: `vegan`, `30-minutes-or-less`, `one-pot` — connecting a Breakfast post and a Dinner post that share a tag, something categories alone could never express.

Navigation Menus

Appearance → Menus lets you build a navigation structure by adding Pages, Posts, Categories, or custom links, then dragging them into the order and nesting you want (nested items become dropdown submenus in most themes).

MENU LOCATIONS ARE DEFINED BY THE THEME, NOT BY YOU

Where a menu can actually appear — a "Primary Menu" in the header, a "Footer Menu" at the bottom — is a set of locations the active theme itself defines, per WordPress Fundamentals 5's own material on themes. Switching to a different theme can mean reassigning your menus to that theme's own, differently-named locations.

Widgets

Widgets are small, self-contained content blocks placed into designated areas a theme provides — commonly a sidebar or a footer — things like a search box, a list of recent posts, a category list, or custom text. In modern WordPress, the Widgets screen itself is built using the same block system introduced in WordPress Fundamentals 4, rather than a separate, older widget interface.

WIDGETS ARE ALSO THEME-DEPENDENT, AND EVOLVING

Which widget areas exist at all is defined by the active theme, exactly like menu locations above. On modern block themes, much of what widgets traditionally did is increasingly handled directly through Full Site Editing's own template parts instead — another concrete instance of the FSE shift previewed in WordPress Fundamentals 4 and 5.

Hands-On Exercises

EXERCISE 1

A cooking blog wants to let readers browse "all vegan recipes" across every meal category (breakfast, dinner, dessert). Explain whether this calls for a category or a tag, and why, using this chapter's own reasoning.

View solution

EXERCISE 2

A site owner switches to a new theme and notices their footer menu has disappeared from the site, even though the menu itself still exists under Appearance → Menus. Explain what most likely happened.

View solution

EXERCISE 3

Explain why this chapter treats alt text as more than an optional metadata field, and describe concretely who is affected if it's left blank.

View solution

CHAPTER 7 QUICK REFERENCE

- The **Media Library** centralizes uploads; alt text matters for accessibility, WordPress auto-generates several image sizes, and a Post's featured image represents it in listings
- **Categories** — broad, hierarchical, structural (table of contents); **Tags** — granular, flat, optional, cross-cutting (index)
- Both apply to Posts only, not Pages, per WordPress Fundamentals 3
- Navigation menus are built under Appearance → Menus, but their actual display locations are defined by the active theme
- Widgets are theme-dependent content blocks, now block-based, and increasingly absorbed into Full Site Editing on modern themes
- **Next chapter:** Users & Roles

CHAPTER 8 OF 10

Users & Roles

WordPress Fundamentals

Chapter 8 · Users & Roles

WordPress Fundamentals 3's own "Pending Review" status was left unexplained — a status that only makes sense once more than one kind of user exists. This chapter introduces exactly that: the role system controlling who can do what on a WordPress site, presented here purely as dashboard behavior. WordPress Intermediate/Advanced 8 revisits every one of these same roles and shows they have real, enforceable teeth in actual PHP code, not just a label in a dropdown.

Why Roles Exist

Any site with more than one person touching content needs different levels of trust — a guest writer shouldn't be able to change site-wide settings, and a content manager shouldn't necessarily be able to create new user accounts. WordPress solves this with five built-in roles, each bundling together a different set of permissions.

The Five Default Roles

ROLE	CAN DO
Administrator	Everything — install plugins/themes, manage all users, change site settings; the role WordPress Fundamentals 2's own install wizard creates automatically
Editor	Publish and manage *every* post on the site (their own and everyone else's), manage categories/tags, moderate comments — but no plugins, themes, users, or settings
Author	Publish and manage only *their own* posts — no access to other users' content
Contributor	Write and edit their own posts, but cannot publish them directly — submitted content sits in the "Pending Review" status from WordPress Fundamentals 3 until an Editor or Administrator approves it
Subscriber	Manage only their own profile — no content creation ability at all; relevant for membership sites or comment systems requiring an account

THE DIRECT CALLBACK TO WORDPRESS FUNDAMENTALS 3

"Pending Review" only exists to serve the Contributor role specifically — every other role either can publish directly (Administrator, Editor, Author) or can't create content at all (Subscriber). One role, one status, purpose-built for each other.

Roles Are Really Just Named Bundles of Capabilities

Underneath the role names, WordPress actually checks individual, granular **capabilities** — `publish_posts`, `edit_others_posts`, `manage_options`, and dozens more — not the role label itself. "Editor" is simply the name WordPress gives to one specific, pre-assembled bundle of these capabilities.

WHERE THIS BECOMES REAL CODE, LATER

This distinction is presented here purely as a mental model. WordPress Intermediate/Advanced 8 shows the actual PHP mechanism — real code checks `current_user_can( 'publish_posts' )`, never "is this user an Editor?" directly — giving this chapter's own role hierarchy its real, code-level enforcement.

Assigning Roles

Users → Add New creates an account with a chosen role from the start; **Users → All Users → Edit** changes an existing user's role. By default, a WordPress user holds exactly one role at a time — though plugins exist that allow finer-grained, custom capability combinations beyond the five defaults.

Choosing the Right Role — A Practical Guide

SITUATION	APPROPRIATE ROLE
A one-person personal blog	Administrator only — no other users needed
An occasional guest blogger, content not yet trusted	Contributor
A hired content manager who shouldn't touch site settings or plugins	Editor
A regular staff writer publishing only their own articles	Author
A community member who just wants to comment with an account	Subscriber

DEFAULT TO THE LEAST ACCESS THAT STILL GETS THE JOB DONE

Granting Administrator access "because it's easier" instead of thinking through what a person actually needs is a genuine, common mistake — the same least-privilege principle covered in real depth in *Database Security*'s own material, applied here to WordPress user accounts specifically. Every account with more access than it needs is one more account whose compromise would do more damage than necessary.

Hands-On Exercises

EXERCISE 1

A site owner hires a freelance writer who should only ever be able to publish their own articles, never touch anyone else's. Explain which role fits, and why an Editor role would grant more access than necessary.

View solution

EXERCISE 2

Explain, using this chapter's own material, why "Pending Review" as a post status would have no purpose on a site with only an Administrator account and no other users.

View solution

EXERCISE 3

Explain what this chapter means by describing roles as "named bundles of capabilities" rather than the actual mechanism WordPress checks, and why this distinction is worth knowing even before seeing any real code.

View solution

CHAPTER 8 QUICK REFERENCE

- **Administrator** — everything; **Editor** — all posts, no settings; **Author** — own posts only; **Contributor** — write but can't publish; **Subscriber** — profile only
- "Pending Review" (WordPress Fundamentals 3) exists specifically to serve the Contributor role
- Roles are named bundles of individual capabilities — WordPress checks capabilities, not role names, in real code (WordPress Intermediate/Advanced 8)
- Assign roles via Users → Add New or Users → All Users → Edit
- Default to the least access that gets the job done — the same least-privilege principle from Database Security, applied to WordPress accounts
- **Next chapter:** Comments & Site Interaction

CHAPTER 9 OF 10

Comments & Site Interaction

WordPress Fundamentals

Chapter 9 · Comments & Site Interaction

WordPress Fundamentals 8's own Subscriber role — an account with no content-creation ability at all — finally has an obvious purpose here: letting a reader comment under a real account rather than anonymously. This chapter covers how WordPress's own built-in comment system works, how to keep it usable despite spam, and the real trade-offs behind common moderation choices.

How Comments Work

A comment form appears beneath Posts by default (Pages can allow comments too, but typically don't by default, matching WordPress Fundamentals 3's own pattern of Pages behaving differently from Posts). Whether comments are open at all — site-wide or per individual post — is controlled from **Settings → Discussion**.

Discussion Settings — The Real Trade-Offs

SETTING	THE REAL TRADE-OFF
Require name/email	A small amount of friction that filters out some low-effort spam and drive-by trolling, at the cost of a slightly higher bar for genuine casual commenters
Require an account to comment	Meaningfully reduces spam and improves comment quality, but genuinely reduces total comment volume — the exact use case WordPress Fundamentals 8's own Subscriber role exists for
Hold a comment for moderation if it contains N+ links	Catches the most common spam pattern (link-stuffed comments) without blocking ordinary comments outright
Automatically close comments after N days	Reduces spam on old posts (which accumulate the most spam attempts over time) at the cost of blocking genuine late discussion

THERE'S NO UNIVERSALLY CORRECT SETTING

A high-engagement community blog and a small business site with occasional posts have genuinely different needs here — more open settings favor engagement and volume; more restrictive settings favor quality and lower

moderation effort. This is a real judgment call, not a checklist with one right answer.

Comment Statuses

Comments move through their own set of statuses, echoing the Post statuses from WordPress Fundamentals 3:

- **Pending** — awaiting moderator approval, not yet publicly visible
- **Approved** — live and publicly visible
- **Spam** — flagged as spam, hidden from visitors, kept temporarily in case of a false positive
- **Trash** — deleted, permanently removed after a set period

Spam — A Genuinely Large Practical Problem

Any public comment form on the internet attracts automated spam almost immediately — WordPress's own popularity, per WordPress Fundamentals 1's own market-share material, makes it a particularly common target for spam tooling built specifically around it.

AKISMET — WORDPRESS'S OWN FIRST-PARTY ANSWER

Akismet, built by Automattic (the same company behind WordPress.com, per WordPress Fundamentals 1's own distinction), checks every submitted comment against a constantly-updated, global database of known spam patterns before it's ever shown publicly. It comes bundled with WordPress core and is, for most sites, the single highest-value anti-spam step available — activating and configuring it is usually the very first real moderation task on a new site.

Engagement — Beyond Just Moderation

A comment section only builds genuine community if someone is actually responding — replying to real comments, not just filtering out spam, is what turns a comment form from a liability into an actual asset. Some sites deliberately choose a third-party comment system (Disqus is a common example) instead of, or alongside, WordPress's own native comments, trading some of WordPress's own direct control for that platform's own spam-filtering and social features — a real option worth knowing exists, even though this course focuses on the native, built-in system.

Hands-On Exercises

EXERCISE 1

A site owner requires visitors to have a registered account before commenting. Explain, using this chapter's own material, what this actually trades off, and which earlier chapter's own material this decision directly connects to.

View solution

EXERCISE 2

Explain why comments flagged as "Spam" aren't deleted immediately, using this chapter's own reasoning.

View solution

EXERCISE 3

Explain why this chapter says there's "no universally correct" discussion setting, using the specific example of a high-engagement community blog versus a small business site.

View solution

CHAPTER 9 QUICK REFERENCE

- Comments are controlled site-wide via Settings → Discussion; Pages typically disable them by default, matching Posts-vs-Pages behavior from WordPress Fundamentals 3
- Every discussion setting is a genuine trade-off between spam reduction and comment volume/engagement — no universally correct answer
- Comment statuses: Pending, Approved, Spam, Trash — echoing Post statuses
- **Akismet** — WordPress's own bundled, first-party spam filter, usually the first real moderation step on a new site
- Requiring an account to comment is the concrete real-world use case for WordPress Fundamentals 8's own Subscriber role
- Third-party comment systems (e.g. Disqus) exist as an alternative to native WordPress comments, trading control for built-in spam handling
- **Next chapter:** Maintenance, Updates & Backups

CHAPTER 10 OF 10

Maintenance, Updates & Backups

WordPress Fundamentals

Chapter 10 · Maintenance, Updates & Backups

A WordPress site is never really "finished" — it's an ongoing responsibility, not a one-time build. This closing chapter covers what keeps a site healthy after launch: updates, backups, and safe ways to test changes before they touch the live site.

Updates — Three Separate Layers

LAYER	WHAT TO KNOW
WordPress core	Minor releases (including security fixes) are applied automatically by default — a genuinely good, safety-first default. Major releases typically require a manual, deliberate update
Themes	Update notifications appear in the dashboard; skipping updates on a heavily-customized theme risks losing compatibility, echoing WordPress Fundamentals 5's own child-theme material
Plugins	Same dashboard-driven process as themes; per WordPress Fundamentals 6, an abandoned or outdated plugin is a real, specific security risk, not just an inconvenience

THIS IS NOT A HYPOTHETICAL RISK

A neglected WordPress update — core, theme, or plugin — is a textbook, real-world instance of *OWASP Top 10*'s own Vulnerable & Outdated Components category: known, publicly documented vulnerabilities in old software versions are actively, automatically scanned for and exploited across the web. WordPress's own huge market share, named back in WordPress Fundamentals 1, makes it a genuinely common, high-value target for exactly this kind of automated scanning.

Safe Update Practice

- **Back up before any major update** — covered in full below
- **Read the changelog** for anything described as a "breaking change" before updating a heavily-customized site
- **Update one thing at a time** when troubleshooting becomes necessary — directly echoing WordPress Fundamentals 6's own plugin-conflict isolation technique

- **Test on staging first**, covered below, for anything beyond a routine minor update

Backup Strategy

A real WordPress backup needs two genuinely separate things, tracing directly back to WordPress Fundamentals 1's own LAMP-stack material:

WHAT	WHY IT'S SEPARATELY NEEDED
The database	Every Post, Page, comment, and user account — the MySQL layer from WordPress Fundamentals 1 — backing up only files would lose all of this entirely
The files	WordPress core, the active theme, every plugin, and the media library — backing up only the database would lose every uploaded image and every installed theme/plugin

WHERE BACKUPS ACTUALLY NEED TO LIVE

A backup stored only on the same server it's protecting is genuinely close to useless — if that server fails entirely or is compromised, the backup is lost right alongside everything it was meant to protect. A real backup strategy stores copies off-site: a separate cloud storage location, a dedicated backup plugin (UpdraftPlus is a common example) configured to push backups elsewhere automatically, or host-level automatic backups that are themselves stored independently of the live server.

Staging Sites

A **staging site** is a private, non-public copy of the live site used to test updates and changes safely before they ever touch the real, live version visitors see. Many hosts (particularly managed WordPress hosts, per WordPress Fundamentals 2's own hosting-type material) offer one-click staging environments built for exactly this purpose.

WHY THIS MATTERS MOST FOR MAJOR UPDATES

Testing a major WordPress core update, a significant theme change, or a plugin known to sometimes cause conflicts on a staging copy first catches problems before they affect real visitors — a genuinely cheap insurance policy against exactly the kind of update-related breakage this chapter has already covered.

WordPress Fundamentals — Where This Course Leaves You

Ten chapters have covered everything a site owner genuinely needs: installation, the dashboard, the Block Editor, themes, plugins, media and organization, users and roles, comments, and now ongoing maintenance — all without writing a single line of PHP. WordPress Intermediate/Advanced picks up exactly where the black

boxes in this course were deliberately left closed — theme anatomy, the template hierarchy, plugin development, and real, code-level security — starting with the exact theme structure WordPress Fundamentals 5 only ever installed as a finished product.

Hands-On Exercises

EXERCISE 1

A site owner backs up only their WordPress database, believing they're now fully protected. Explain what they would still lose in the event of a server failure, using this chapter's own material.

View solution

EXERCISE 2

Explain why this chapter connects a neglected plugin update directly to a real, named OWASP Top 10 category rather than describing it only as a vague "best practice."

View solution

EXERCISE 3

Explain why storing a backup only on the same server it protects is described as "genuinely close to useless," and describe what a correct backup location looks like instead.

View solution

CHAPTER 10 QUICK REFERENCE — COURSE COMPLETE

- Core minor/security updates apply automatically by default; major core, theme, and plugin updates need deliberate action
- A neglected update is a real instance of OWASP Top 10's own Vulnerable & Outdated Components category, not a hypothetical risk
- Safe updates: back up first, read changelogs, update one thing at a time when troubleshooting, test on staging for anything major
- A real backup needs both the **database** (all content) and the **files** (core, theme, plugins, media) — and must live off-site to be genuinely useful
- **Staging sites** — a private copy of the live site for safely testing changes before they go live

- This completes WordPress Fundamentals — WordPress Intermediate/Advanced opens every black box this course deliberately left closed