Reading This Handbook

- Requirements
- Your next steps
  - For newcomers
  - For seasoned theme authors
  - For classic themes documentation
- How to read code examples
  - Namespacing examples

The goal of this handbook is to walk you through the basics of modern WordPress theme development in its early chapters. Then, work through more advanced topics with each chapter that follows.

The quickest way to learn theme development is to simply read this handbook from beginning to end and follow along with its examples.

Requirements

The biggest requirement for building a WordPress theme is a willingness to learn. You are here reading this handbook, so let’s check that one off the list of requirements.

WordPress theming has changed many times over the years. Today, the pathway for creating one is much lower than it was in the past. You can absolutely create a theme with no coding knowledge. But you will find it much easier to familiarize yourself with a few web languages.

You will see HTML, CSS, PHP, JSON, and JavaScript within the handbook, so it helps to be able to easily recognize what language you are looking at. HTML and CSS are foundational pieces of the web, so those should be prioritized over others.

The following are external resources that you can use to learn more, but there are 1,000s of guides and tutorials around the web:

- MDN Web Docs: HTML
- MDN Web Docs: CSS

- PHP official documentation
- MDN Web Docs: JSON
- MDN Web Docs: JavaScript

You can get pretty far into theme building via the WordPress user interface, at least by modifying and exporting an existing block theme. But you'll want to pick up a few more skills to build more advanced features, or to truly create a theme from scratch.

Your next steps

For newcomers

It is time to truly embark on your journey into learning theme development. It is an exciting moment, so get ready for an adventure.

Setting things up: if this is your first time building a theme, or if you just want a refresher on the basics of theme development, your next stop should be the Tools and Setup page. This will help you set up your development environment and determine which tools you need to start off on the right foot.

Getting off to a quick start: once you're set up, check out the Quick-start guide, which is aimed at giving you a taste of what's to come. It will walk you through setting up a theme to work with.

Understanding the basics: once you have everything set up and running, move onto the Core Concepts chapter. There, you will learn the foundational principles behind creating WordPress themes.

Afterward, just keep working through each chapter of the handbook.

For seasoned theme authors

Even if you've been creating themes for years, it never hurts to give the full handbook a reading from time to time. It is regularly updated, so there may be new content that you haven't seen before.

Feel free to hop around to find the specific topic you need using the handbook's navigation links. Whatever the case, the Theme Handbook is here to help you discover a solution.

Feeling pretty hardcore? Jump over to the Advanced Topics chapter.

Can't remember the naming format for a particular template? Find it in the Template Hierarchy docs.

Need to brush up on `theme.json`*?* No problem. Check out the Global Settings and Styles documentation.

You can also find theming tutorials and walk-throughs under the Themes category on the WordPress Developer blog if you can't find the topic you're looking for in the handbook.

For classic themes documentation

While some of the content in most chapters will apply to classic themes, the docs are primarily geared toward modern block theming. However, there is a dedicated Classic Themes chapter if you need to locate documentation on a specific classic feature.

How to read code examples

Throughout the handbook, you will see code examples, and this section is aimed at helping you understand how to read them.

The handbook will use a consistent "namespace" or "prefix" throughout its pages, generally seen as `theme_slug` or `theme-slug`. This namespace is meant to be replaced in your theme with one that is unique to it.

But how do you know what your namespace is? The most straightforward way is to use your theme's name. If your theme is titled **Fabled Sunset**, you would convert it to the appropriate format for the use case. For example, it would become `fabled_sunset` when used in a PHP function name or `fabled-sunset` as a handle/slug/ID.

The practice of "namespacing" code is decades old. Essentially, a namespace is an identifier for a group of objects created so that they do not conflict with objects in other namespaces.

WordPress has a vast ecosystem of plugins and themes. Each needs a way to distinguish itself from the others. Without unique namespaces, it would get a bit chaotic and errors would arise.

For example, if your theme declared a custom function named `get_post()` instead of `fabled_sunset_get_post()`, your site would fail with a fatal error because WordPress already has a function with this name:

Screenshot of an error message stating that you cannot redeclare the get_post() function, whi

Avoiding these types of errors is the primary reason the handbook will include namespaced or prefixed code examples.

Namespacing examples

Using the fictional theme name **Fabled Sunset**, let's look at some examples of formatting your namespace. Don't worry if you don't understand these yet or if you do not know anything about code. This is only meant to help you read the handbook.

Usage in script and style handles:

```
<?php
// theme-slug-editor becomes:
wp_enqueue_script( 'fabled-sunset-editor', ... );

// theme-slug-main becomes:
wp_enqueue_style( 'fabled-sunset-main', ... );
```

Usage in text domains (used for translations):

```
<?php
// theme-slug becomes:
esc_html_e( 'Hello, world!', 'fabled-sunset' );
```

Usage in PHP Functions:

```
<?php
// theme_slug_func() becomes:
function fabled_sunset_func() {
    // ...
}
```

Usage in PHP classes:

```
<?php
// Theme_Slug_Class() becomes:
class Fabled_Sunset_Class() {
    // ...
}
```

As you read through the handbook, you will learn when to use each case. For now, just know that you should always replace the example namespace with one that is unique to your project.

The WordPress Coding Standards encourages the use of PHP’s built-in namespaces. The same guideline would apply: use your theme’s name to create your namespace. In this case, the namespace for the example theme would be `Fabled_Sunset` or `FabledSunset`. The Theme Handbook uses a prefixed style for PHP function and class names, as shown above. This is primarily because it is simpler for showing one-off code examples.

□□□ #2
□ □□□ □□□ 2026-10-31 18:35:11 UTC
□ □□□ □□□ 2026-10-31 18:35:13 UTC