WORDPRESS THEMES IN DEPTH

COMPLETE GUIDE TO BUILDING AWESOME THEMES

WP 6.2

JEFF STARR

MAP

Welcome

Overview

Theme Anatomy

Exploring Template Files

Customizing WordPress Themes

WordPress Theme Development

Security, Optimization & Testing

Front-End Techniques

Advanced Functionality

Sharing & Selling Themes

Build a Starter Theme

2020 Theme Walkthrough

Appendix

About this Demo

This is a PDF demo of the book, *WordPress Themes In Depth*. It contains the entire Table of Contents and the first few pages of each chapter. The complete book contains over 450 pages and is packed with 100% theme-building goodness.

Note that some hyperlinks may be disabled in this demo PDF. In the complete version of the book, however, all hyperlinks are fully functional and ready to rock.

Welcome

1
Welcome

Introduction

You are about to embark on an in-depth journey through the vast ocean of WordPress theme development. On this voyage you will learn everything needed to customize, build, debug, optimize, secure, share, and even sell your own WordPress themes. You'll learn how themes work, and how a well-built theme taps WordPress' wealth of functionality to create a feature-rich interface that delivers the best possible user-experience.

WordPress theme development brings together many aspects of web design, including numerous coding languages, SEO strategy, web security, performance optimization, and everything in between. But as we'll see, you don't need to be an expert in every field to build your own awesome themes. Rather, like a chef using prepared ingredients to create delicious meals, you can apply prescribed techniques to build awesome themes[1].

Whether you are just getting started in web design or have years of experience under your belt, building and customizing your own WordPress themes can be challenging, rewarding, and lucrative. There is much to learn, and with this book as your guide, you are on your way to mastering the craft and creating your own amazing themes. So welcome aboard, may your journey be filled with adventure, discovery, and success.

1. This book focuses on classic theme-building. Classic theme building makes use of PHP templates, HTML, CSS, and some JavaScript. Most themes are built this way.

Newer (Gutenberg) versions of WordPress also support a more JavaScript-oriented way to build themes using "blocks". Learn more about Block Themes at WP.org.

In the beginning

In the beginning, WordPress was new and themes were few. As WordPress grew in popularity, new and exciting themes began to flourish. Designers and developers began pushing the limits of WordPress and sharing their own creative and unique themes with others in the community.

Eventually, WordPress reached critical mass and the premium theme market erupted on the scene. Premium themes boasted awesome features, more functionality, and better customer support. Since around 2007, the ocean of WordPress themes — both free and premium — has grown wide and deep, providing millions of WordPress users with a growing abundance of truly amazing themes.

Ever since, WordPress themes have continued to multiply, evolve, and integrate new technologies and design techniques. Today's themes are diverse, niche-oriented, and loaded with features. Today's themes are built to be responsive, so they look great regardless of browser or viewing device. And tomorrow's themes will be even better, with more people building themes and using WordPress than ever before.

The theme market remains fierce, but there is an abundance of WordPress users and a wealth of opportunity. With the right mindset and this book as a guide, you're equipped to customize and build your own awesome themes to use, share, and rock the marketplace.

One of the first default themes, Kubrick was included in many versions of WordPress from around 2005 to 2010. The above screenshot shows Kubrick version 1.5 working "out of the box" in WordPress 4.0.

Goals of the book

The primary goal of this book is to build, customize, and sell your own WordPress themes. To achieve that goal, the book provides the following information:

- **Chapter 1** provides a concise overview of the book.
- **Chapters 2–4** explain how themes work and how to get started with WordPress theme development.
- **Chapter 5** explains how to customize WordPress themes.
- **Chapters 6–10** dive deep into theme development, security, testing, and optimization.
- **Chapter 9** provides extensive coverage of advanced techniques, including Theme Features, Custom Fields, Theme Options, and much more.
- **Chapter 10** explains everything you need to know to share and sell your own WordPress themes.
- **Chapter 11** provides complete steps for building your own lightweight and flexible starter theme.
- **Chapter 12** walkthrough of the 2020 theme, which is a premium theme that's modular, responsive, and full-featured.
- The **Appendix** contains information on getting help, troubleshooting, debugging, resources, and more.

This book is organized to help you study WordPress themes in depth. In general, the book presents information in cumulative fashion, with each Chapter building on previous material.

As mentioned, theme development brings together many disciplines and coding languages. To help keep things as simple as possible, the book stays focused on theme-building without drifting too far into any one aspect, such as CSS or JavaScript. Throughout the book we will be using CSS, HTML, JavaScript, and PHP, but it's mostly copy-&-paste followed by explanations. Simplicity and clarity are the key.

Where to begin?

Depending on your current level of experience, you may want to jump in at some specific point in the book. To help with this, here is a quick overview that highlights some of the book's key areas.

Theme fundamentals	Chapters 2–4
Customizing themes	Chapter 5
Theme development	Chapter 6
Build your first theme	Chapter 3 – Simplest Theme
Build a generalized theme	Chapter 6 – General Theme
Build a starter theme	Chapter 11 – DIY Theme
Premium theme walkthrough	Chapter 12 – 2020 Theme
Front-end techniques	Chapter 8
Advanced Techniques	Chapter 9
Sharing and selling themes	Chapter 10
Securing themes	Chapter 7
Optimizing theme	Chapter 7
Testing themes	Chapter 7

New to WordPress themes? Check out the official guide for new users:

https://wordpress.org/support/article/using-themes/


Overview

2
Overview

What is a WordPress theme?

According to WordPress.org[1], the latest version of WordPress has been downloaded nearly 30 million times. While not every download develops into a full-fledged website, the ones that do are going to need at least one WordPress theme. That presents a huge amount of potential for theme designers, developers, and marketplaces.

So what exactly is a WordPress theme? Technically, a WordPress theme is nothing more than a collection of files that work together to display content to the user. WordPress itself ships with three of its own default themes. These are meant to showcase the diversity and potential of themes, while also enabling the user to customize how their site appears to visitors on the front-end.

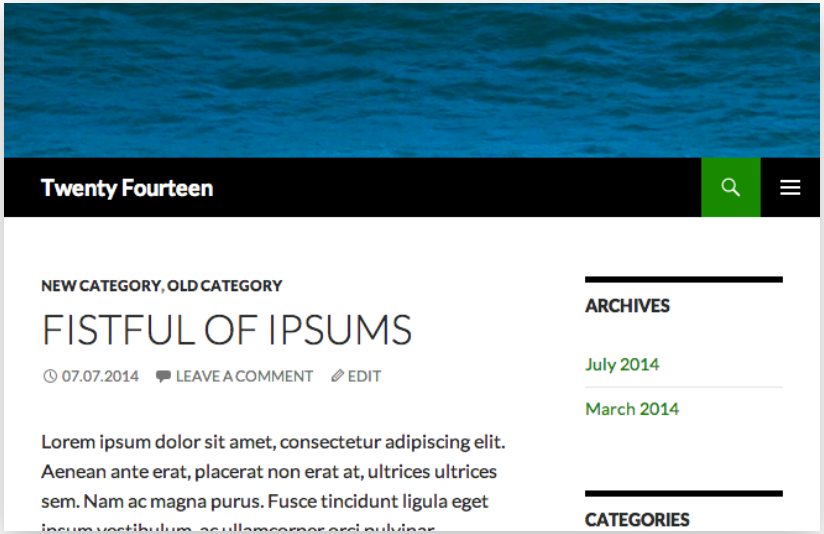
Themes enable users to tap into the power of WordPress to display and manage their content exactly as desired. As we'll see, there are themes for just about every niche and clique imaginable. Users can download themes for blogs, shops, portfolios, magazines, and so much more. Indeed, themes bring together the power of WordPress with the creativity of designers, developers, and savvy users.

Once installed and activated, themes may be customized in numerous ways. Most of the newer, cutting-edge themes ship with loads of options and features, enabling users to dial in the perfect theme without touching a line of code. Other themes, such as starter

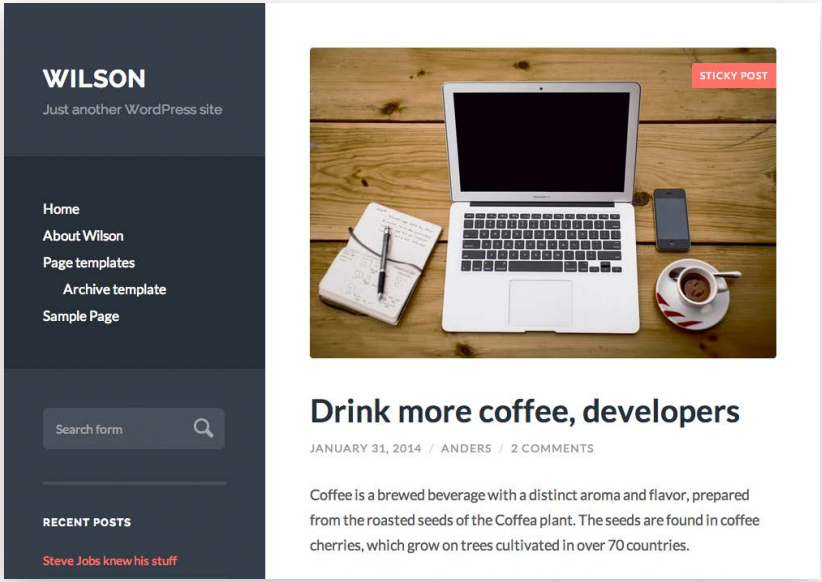
1. WordPress download counter: https://wordpress.org/download/counter/

templates and simple themes, may include no options whatsoever. So in order to customize things, it may be necessary to work directly with template files. Either way, themes provide full control over how content is displayed on the front-end. In terms of customizing site appearance and functionality, the theme is where it all happens.

To give you a sense of what's possible, this chapter showcases some awesome themes from the WordPress Theme Directory. As we'll see, themes can be very general like the typical default themes, or very specific like a real-estate theme. Themes can be very simple, using as few as two template files, or they can be very complex, requiring hundreds of files. The Twenty Fourteen WordPress theme, for example, is a magazine-style theme consisting of around 70 files.



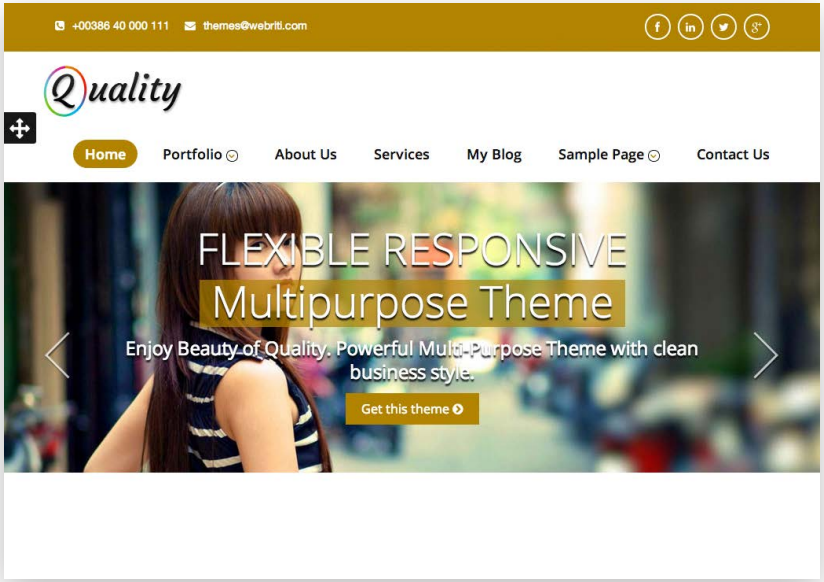
Twenty Fourteen is a responsive magazine-style theme with a sleek, modern design. (Included with WordPress)



The Wilson theme is one of my favorites. Clean, simple, and responsive. Check it out at the Theme Directory. https://wordpress.org/themes/wilson/

Further, consider that WordPress-powered sites are dynamic, meaning that content is stored in a database and displayed on the front-end based on the active theme's template code. In other words, the theme provides a user-interface for accessing database content.

Themes also enable customization of the WordPress Admin Area. For example, the 2020 theme includes a set of functions that customize various aspects of the Admin Area, such as the Login Page, Dashboard, Toolbar, and footer text. Although modifying the Admin Area is best left to plugins, it's nice to know that themes can do it too.



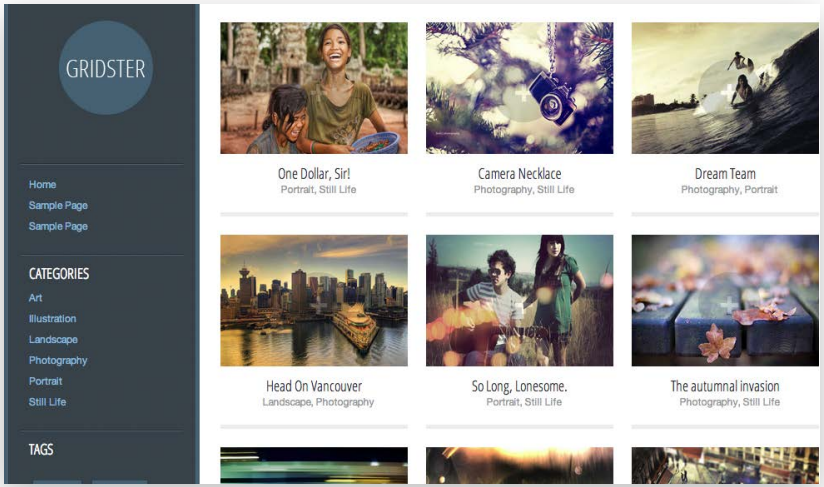
Quality is a stylish theme designed for corporate and business sites. Get it at the Theme Directory. https://wordpress.org/themes/quality/

WordPress themes have access to all of the functionality provided by WordPress. This means that we can tap into WordPress and use the functions that are needed to display our content as desired. So WordPress provides the functionality, the database stores the content, plugins extend and enhance WordPress, and themes bring it all together with a flexible template system.

Another important thing that themes do is separate the user-interface (UI) from the core WordPress files. This degree of separation enables you to load up on functionality without touching any of the core files. This means that sites can update the WordPress core without changing anything in the theme, so millions of users to stay current with the latest and greatest. Indeed, themes play an important role on the Web, enabling users to make the most of WordPress.

What can themes do?

As we've seen, themes play a key role by bringing together a site's content, design, and functionality. In doing so, themes tap into WordPress' vast arsenal of functions, hooks, and tags to generate a UI for interacting with the database. Essentially, themes enable you to harness the power of WordPress to create a unique, highly customized design that suits your site's purpose, scope, and goals.



Gridster is a responsive portfolio theme for photographers that features a nice grid-based homepage. https://wordpress.org/themes/gridster-lite/


Theme Anatomy

3
Theme Anatomy

Diving in

Now that we're set up to work with themes and have a general understanding of what they do and how they work, it's time to dive in with a in-depth look at theme anatomy. In this chapter, you will learn about the anatomy of a typical WordPress theme. WordPress themes may vary greatly in terms of functionality and features, but the underlying concepts and techniques remain consistent throughout all well-built themes. Understanding core concepts such as Page Views, the Template Hierarchy, and the WP Loop will enable us to customize and build our own awesome themes.

Additionally, we'll review the basics of using various coding languages such as CSS, HTML, and PHP. Along the way, we'll look at some examples, apply some techniques, and prepare to go further into WordPress themes in the next chapter.

As you go through this chapter and the remainder of the book, it can be very beneficial to follow along with examples and techniques. If possible, crack open your favorite code editor and browser to get some practical, hands-on experience.

Understanding Page Views

Page Views play an important role in how WordPress themes operate. In WordPress, the Page View refers to the type of web page generated by WordPress. For example, when viewing a single blog post, you're

WordPress Page Views

Here is a quick chart showing the default WP Page Views along with an example URL for each. For all example URLs, remember to change the domain name and paths to match your site.

Note that we're using the default permalink structure here. If a custom permalink structure is enabled, these default URLs will work, but will be redirected to their new permalink URLs.

Another thing to keep in mind is the example URLs are just that – examples of URLs that, when entered in a browser, will lead to the associated type of page-view. Note that most of these page views are accessible via multiple URLs.

Page View	Example URL
Home Page (page 1)	`https://example.com/`
Home Page (page 2, 3, ...)	`https://example.com/?paged=2`
Single Post	`https://example.com/?p=1`
Single Page	`https://example.com/?page_id=2`
Tag Archive	`https://example.com/?tag=test`
Category Archive	`https://example.com/?cat=1`
Author Archive	`https://example.com/?author=1`
Date Archive (year)	`https://example.com/?m=2014`
Date Archive (year/month)	`https://example.com/?m=201404`
Date Archive (year/month/day)	`https://example.com/?m=20140407`
RSS Feed	`https://example.com/?feed=rss2`
404 (Not Found)	`https://example.com/dfgdfgdfgdg`
Search Results	`https://example.com/?s=searchterm`

looking at a "single view" or "single-post view". Likewise, when viewing the archives, you're looking at an "archive view". It's a simple yet important concept that is key to understanding how themes work on the front-end. To get a better idea of how it works, check out the popout box, "WordPress Page Views", on this page.

As seen in the chart above, Page Views are associated with specific URLs. For example, to access a single-post view, we could visit `https://example.com/?p=1` in a browser. If no content is found for the requested URL, WordPress delivers a 404 (Not Found) page.

Every theme displays *something* for each type of page view. What that "something" is depends on the requested page and which template files are included in the theme.

For example, if an author-archive view is requested, WordPress checks the theme to see which template files are available to display the page. In general, the process goes like this:

- WordPress first checks for an author-specific template to generate the requested page.
- If an author-specific template is not found, WordPress checks for an archive-specific template.
- Lastly, if no archive-specific template is found, WordPress generates the page using the default template.

This process of using the most specific template to display the requested page is referred to as the Template Hierarchy.

The Template Hierarchy

WordPress' Template Hierarchy refers to the order in which template files are used to display various types of web pages. Each theme may have a completely different set of template files, but WordPress will always use the Template Hierarchy to determine which files are used to render the requested page. WordPress themes may include as many or as few different template files as needed depending on the design. For example, a complex theme may include a distinct template for each type of Page View, while a minimal theme may use only two or three templates to handle all of them. It's entirely up to the theme designer to determine which templates are necessary based on the Template Hierarchy.

Before getting into theme template files, let's take a look at a simplified overview of Template Hierarchy shown on this page. There we see how the hierarchy applies to different page views. For example, when a single-post view is requested, WordPress first checks for a template file named `single.php`. If that's not available, WordPress generates the page based on `index.php`, which is one of the two default template files required for any theme.

✳ The simplest theme

To build the simplest possible theme, we need include only two files:

- `index.php`
- `style.css`

WordPress Template Hierarchy

Here is an overview of the Template Hierarchy[1-2]. It includes the most commonly used template files and their associated Page Views. For a more complete view of the Template Hierarchy, visit the popout box, "Which template file will wordpress use?" in the previous chapter.

Page View	Theme Template		
	if exists	**else**	**else**
Home Page (posts)	`home.php`		`index.php`
Home Page (page)	`front-page.php`		`index.php`
Single Post	`single.php`	`singular.php`	`index.php`
Single Page	`page.php`	`singular.php`	`index.php`
404 (Not Found)	`404.php`		`index.php`
Tag Archive	`tag.php`	`archive.php`	`index.php`
Category Archive	`category.php`	`archive.php`	`index.php`
Author Archive	`author.php`	`archive.php`	`index.php`
Date Archive	`archive.php`	`archive.php`	`index.php`
Search Results	`search.php`	`archive.php`	`index.php`

This simplified look at the hierarchy shows some of the most commonly seen template files and their associated page views. Also common, but not shown in the chart, are custom template files for posts, pages, and archives. For example, you can create a custom page template for your "About" page by creating a file named `page-about.php`.

1. Official Documentation: `https://wp-tao.com/678`
2. For all of the gory details, check out this infographic: `https://wp-tao.com/679`

Exploring Template Files

4
Exploring Template Files

Welcome to the tour

In this chapter, we take a complete tour of a "typical" WordPress theme. Good themes may vary wildly in terms of which files are included and which code is used, but they all employ the same general principles and should adhere as closely as possible to the WordPress API. So while the number of contents of various template files may vary, most themes share enough common elements to make possible a "tour" of template files included in the typical theme.

As we look at each file, we'll consider its general purpose, structure, and contents. Along the way, we'll examine some key techniques for modifying and enhancing default functionality. While staying focused on the code, we'll also look at how each template file is displayed on the site's front-end, highlighting cool features and seeing how it all fits together. In this chapter, we'll be exploring the following template files:

- `style.css`
- `index.php`
- `header.php`
- `sidebar.php`
- `footer.php`
- `single.php`
- `page.php`
- `comments.php`
- `functions.php`
- `content.php`
- `searchform.php`
- `front-page.php`
- `home.php`
- `archive.php`
- `attachment.php`
- `author.php`
- `category.php`
- `date.php`
- `search.php`
- `tag.php`
- `404.php`
- *+ misc. files*

Additionally, this chapter covers custom template files for custom post types, custom taxonomies, and more. Other types of files are discussed briefly near the end of the chapter, where we look at JavaScript, images, and other commonly included files.

Think of this chapter as a general walkthrough of the typical WordPress theme. It's also very complete, so you could follow along and use the information provided in each section to assemble a fully functional theme. Doing so would serve as an excellent exercise for learning how to build your own themes, but the end result would be a rather lean and general looking theme. But no worries — later in the book, we step through the process of building a complete starter theme, and walk through the features of 2020, a premium theme that is fully loaded with tons of functionality and features.

So with that in mind, let's begin the tour!

style.css

`style.css` is the theme's main stylesheet. It is required for a theme to be recognized by WordPress, and it must always be included in the theme's root directory. Further, `style.css` must always include a proper file header[1]. The `style.css` header includes the theme's name, description, author, license, and other essential information.

A typical `style.css` header looks like this:

```
/*
   Theme Name: General Theme
   Theme URI: https://example.com/general-theme/
   Author: Theme Designer
   Author URI: https://example.com/
   Description: A hypothethetical, generalized typical theme
   Version: 1.0
   License: GNU General Public License v2 or later
   License URI: https://www.gnu.org/licenses/gpl-2.0.html
   Tags: black, white, light, dark, two-columns
   Text Domain: generaltheme
   Domain Path:  /languages
*/
```

This file header is included as an inline comment. When writing CSS, inline comments give designers a place to include non-CSS content in the stylesheet. So, for example:

```
/* this is a single-line inline comment */
```

And likewise:

```
/*
       this is an inline comment
       that spans multiple lines
*/
```

If you look through the `style.css` file of WordPress' Twenty Fourteen theme, you will find inline comments used to organize the code and convey useful bits of information.

1. File Headers explained: https://codex.wordpress.org/File_Header

In general, the `style.css` header should include as much information as possible, however, technically, only the Theme Name is required in order for WordPress to recognize and utilize the theme. Other header information is optional, but should be included if the theme is to be made available to the public. We'll learn more about this in Chapter 10, where we learn how to prepare our themes for public use. For the sake of completeness, here is a quick explanation of each line in the `style.css` header, along with any relevant details.

- Theme Name – the name of the theme, can be anything, but must be unique among other installed themes
- Theme URI – the online location of the theme's homepage
- Author – the name of the author, should be the author name, or the author's username if registered at wordpress.org
- Author URI – the online location of the author's homepage
- Description – a clear, succinct description of the theme
- Version – the version number of the theme
- License – the name/summary/link of the theme license
- License URI – the online location of the license
- Tags – tags used to describe the theme, also used to help users find new themes when searching the WP Theme Directory[1]
- Text Domain – the theme's text domain, used in localization and theme translation[2]
- Domain Path – the path to the theme's localization files, relative to the themes directory[3]

As you get started in theme development, don't worry too much about including all of these details. Include as much information as possible to get going, and then revisit as needed to supply additional details. For example, it's probably not necessary to include international language support in your first theme, so it's fine to exclude the Text Domain and Domain Path information. Then, once you get to the point where you want to share or sell your own themes, read through Chapter 10 to learn how to hook everything up and provide a complete `style.css` header.

One more note about the `style.css` header, it's fine to include additional information as well. For example, in the Twenty Fourteen WordPress theme, the `style.css` header includes the following additional information:

```
/*
...

This theme, like WordPress, is licensed under the GPL.
Use it to make something cool, have fun, and share what you've
learned with others.
*/
```

That's just free-floating information included at the bottom of the style header; it could have been 30 paragraphs of text and it would still be fine. WordPress only cares about the information associated with the keywords defined in the previous list.

1. Tags are used to filter themes in the Theme Directory, for example: https://wordpress.org/themes/tags/grid-layout/
2. See Translation Support in Chapter 10, and learn more at the Codex: https://codex.wordpress.org/I18n_for_WordPress_Developers
3. So, for example, `/languages` means that the directory is located in `/general-theme/languages/`


Customizing WordPress Themes

5
Customizing WordPress Themes

Anything & everything

Perhaps the greatest thing about WordPress is its flexibility. Regardless of your skill level, there are a million ways to customize WordPress to create the perfect site. This chapter is for anyone who wants to customize their theme. It explains all the main ways of doing so, provides a wealth of examples, and broadens our perspective of how the theme fits into the grand scheme of things.

With WordPress, not only is it possible to customize virtually any aspect of your site, there are myriad ways of doing so. For example, just looking at the sidebar, we can customize using widgets, the style template, the sidebar template, or a child theme. It's also possible to make changes via the functions template or a plugin, depending on which strategy makes the most sense.

As you read through this chapter, it's a good idea to follow along with one of the demo themes or one of the default WordPress themes. Don't hesitate to experiment with different techniques and ideas — it's impossible to permanently "break" anything if you have current backups of your database and files[1].

One more note before diving in, some of the techniques in this chapter are fine to apply to a live site. For example, it's fine to customize widgets on a live site. Other techniques should be implemented while the site is not accessible to the public[2].

1. See Setting up to work with themes in Chapter 2.
2. See Local vs. live in Chapter 2.

Many ways to customize

There a million and one ways to customize WordPress. But you don't want to just jump in and start changing things without a plan. First, make sure you understand what you are doing. Whether it's a widget, plugin, or custom function, make sure you're paying attention, especially when making changes on a live site. It's fine if you don't understand the precise mechanism behind every snippet of code, so long as you're getting your information from a reliable source.

A good strategy for customizing themes is to keep it simple and begin with the least invasive, most applicable method. Whenever possible, use WordPress' native settings and features to customize your site. When that's not possible, see if there is a widget or plugin that will do the job. When none of these methods suffice, you can customize the theme template, either directly or via Child Theme.

To put things into perspective, here is an overview of the customization methods that are covered in this chapter[1].

✳ Theme settings

Most themes these days include a Theme Options screen in the Admin Area. If so, this should be the first stop for customizing your theme. As much as possible, any customization should be accomplished through the Theme Options screen.

✳ Theme Customizer

The Theme Customizer makes it simple to customize various aspects of your theme. Themes vary, however, in terms of how well the Customizer is supported. Many themes provide options for changing things like titles, taglines, and colors, but other theme features like custom backgrounds and custom headers require theme support. If in doubt, check the theme's documentation.

✳ Plugins & widgets

As we'll see in the next chapter, plugins are an excellent way to add, change, or remove existing WordPress functionality. For example, if a theme lacks a built-in contact form, simply install one of the many contact-form plugins and move on with your life.

✳ Shortcodes

As we've seen, shortcodes make it easy to include custom snippets with your posts and pages. If your theme includes shortcodes, they may enable the required customization. New shortcodes may be added directly via `functions.php`, or from a theme or plugin.

✳ Child themes

The next best way to make changes to your theme is using a child theme. As discussed in the previous section, child themes enable you to overwrite any part of the theme with custom functionality. Customizing via child theme is smart because the parent theme's template files remain unmodified.

1. WordPress 5+ features a new post editor. So now there is a Block Editor (aka "Gutenberg"), that replaces the original Classic Editor. Here is a round-up of useful resources to learn more about the Gutenberg Block Editor: https://wp-tao.com/667

Which method should I use to customize my theme?

Method	Things that can be customized	Difficulty
Admin Area & default WordPress settings	Basic functionality such as enabling/disabling comments, the number of posts to display, enabling/disabling registration, setting up permalinks, etc.	Easiest.
Install & configure themes/plugins	Everything under the sun, but some niches/functionality are hit or miss. Plus, some plugins may require advanced configuration.	Easy to Advanced (usually easy)
Configure theme options/ widgets	Colors, font-sizes, positioning of sidebars, header images, background images, menus, widgets, colors, fonts, and potentially much more, depending on the theme.	Easy.
Customize theme template files	Page structure (HTML), appearance (CSS), behavior (JavaScript), and dynamic functionality (PHP). Essentially, anything is possible via template customization.	Medium/Advanced (varies with theme)
Edit some WordPress core files..	Literally any aspect of WordPress. As a rule, you should never change any core files on a production site. There is always a better way.	Never do this.

✳ New theme

Another option for changing your site's design is to find a new theme. There are so many great themes available, it's generally easy to find something that fits the bill. Visit Chapter 2 for more information about different types of themes and how to find good themes.

✳ Template files

It's also possible to make changes to the theme itself. By editing a theme's template files, you're able to customize literally anything you wish. So if there's something that needs customizing, and it's not possible to do so using any of the methods above, editing the template files should get you there.


WordPress Theme Development

6

WordPress Theme Development

Before diving in

To put things into perspective, WordPress theme development is just a small part of the much larger field of dynamic web development. Anything that can be built with regular coding languages like PHP, HTML, JavaScript, and CSS, also can be built with WordPress. Of course, the more you know about these languages the easier it's going to be to learn how to build WordPress themes, but there is no need to be an expert in everything. In this chapter, you will see how easy it is to get started with WordPress theme development, and learn the principles and methodology needed to go further.

Before diving in, it's important to know exactly where we're at and where we're going. At this point in the book, we have a solid understanding of WordPress theme structure and how template files are used to display content on the front-end. We're now ready to apply this information and build some amazing themes.

To get there, we'll need to stay focused on the theme itself, without veering too far into any particular aspect of web design. It would be easy to get sidetracked on, say, the graphic design aspect of web design. Or spend an entire chapter on the latest UX/UI wizadry. Or go chapters deep into mobile design and advanced JavaScript techniques. Instead, we're going to keep things simple and focus on building awesome themes.

As we go, keep in mind that there are many ways to go about building WordPress themes — there is no "one right way" to do it. Building

themes is like playing music. Yes you can just play the notes, but the results will be plain and ordinary. To really rock a WordPress theme, you need to find your own rhythm, and that requires understanding and lots and lots of practice. This book provides the understanding, and this chapter provides the fundamentals so you can dive in and start practicing.

Build a theme in 4 steps

Before doing anything, let's see how easy it is to build a simple WordPress theme in four simple steps:

Step 1

In the WordPress `/themes/` directory, create a folder named `/simplest-theme/`

Step 2

In the new directory, create two new (empty) files, `style.css` and `index.php`

Step 3

Open `style.css` and add the following code:

```
/* Theme Name: Simplest Theme */
```

Step 4

Open `index.php` and add the code provided in the next column.

Congratulations, you just built a WordPress theme.

```
<!DOCTYPE html>
<html>
   <head>
      <title><?php bloginfo('name'); ?><?php wp_title(); ?></title>
      <link rel="stylesheet" href="<?php bloginfo('stylesheet_url'); ?>">
   </head>
   <body>
      <header>
         <h1><a href="<?php echo home_url('/'); ?>">
            <?php bloginfo('name'); ?></a>
         </h1>
         <p><?php bloginfo('description'); ?></p>
      </header>
      <article>
         <?php if (have_posts()) : ?>
            <?php while (have_posts()) : the_post(); ?>

            <div class="post">
               <h2><a href="<?php the_permalink(); ?>">
                  <?php the_title(); ?></a>
               </h2>
               <?php the_content(); ?>
            </div>

            <?php endwhile; ?>

            <div class="post-nav">
               <?php posts_nav_link(); ?>
            </div>
         <?php endif; ?>
      </article>
      <footer>
         <p>&copy; <?php echo date('Y'); bloginfo('name'); ?></p>
      </footer>
   </body>
</html>
```

Yes, it is a simple theme, but it is suitable for displaying content for every type of page view (e.g., single posts, archives, etc.).

If you followed along with the four steps, go ahead and install and activate the "Simplest Theme". Then surf around the front-end of your site and check out the various page views to see how things look with a theme that weighs under 12 KB. For example, if your site is located at `https://example.com/`, try visiting the following URLs:

URL	Page View
https://example.com/?p=1	Single Post
https://example.com/?page_id=2	Single Page
https://example.com/?paged=2	Paged view of Home Page
https://example.com/?cat=1	Category Archive
https://example.com/?author=1	Author Archive
https://example.com/?m=2014	Year Archive
https://example.com/?m=201401	Month Archive
https://example.com/?m=20140101	Day Archive
https://example.com/?s=hello	Search Results
https://example.com/404page	404 Not Found

As you can see, less than 40 lines of basic code gets you a fully functional theme that displays content for every type of page view. Sure, without any CSS, the markup looks plain and boring, but the Simplest Theme illustrates some key concepts:

✳ The Loop is versatile

One default loop in the `index.php` is "smart" enough to display the correct content for each type of page view. It doesn't matter how many template files a theme might have — one or many — the Loop always knows exactly which posts to display.

✳ Diversity of page views

Using a simple template file (i.e., `index.php`) to display all page views means that all page views are going to look virtually identical. As discussed in depth in Chapter 3, the Template Hierarchy enables us to customize the appearance of different page views, so search results can look like search results, Pages can look like Pages, and so forth.

✳ Required tags

Are there any *required* tags for WordPress themes? The short answer is "no", but it also depends on how the theme will be used. For personal themes that nobody else will use, you can knock yourself out and use as many or as few template tags as desired. But if you're planning on distributing your theme for others to use, then the answer is "yes", and there are a number of template tags that should be included[1].

1. See Theme Guidelines Checklist in Chapter 10.


Security, Optimization & Testing

7

Security, Optimization & Testing

It's all about WordPress

The *good news* is that securing, optimizing, and testing themes is much easier than doing the same for an entire installation of WordPress. The *not-so-good news* is that a lot of work may be required to secure and optimize things at the theme level. Of course, the amount of work required depends on the theme. Themes that use the WP API whenever possible tend to require less work than themes that go it alone with their own custom code.

This is why, throughout the book, we stick close to the WP API in every step. Whenever possible, we let WordPress do the heavy lifting and keep our template code as clean as possible. This way, we are baking security, performance, and SEO right into the theme, rather than tacking it on at the end. In this chapter, we dive into theme security and optimization, and learn how to test many different aspects of WordPress themes.

In general, security, performance, and SEO work together to determine the overall quality and success of a website. Search engines such as Google determine site quality by measuring signals of quality like performance, relevancy, reputation, and many other factors[1–2]. In this chapter, you will learn how to optimize for these factors at the theme level, where good organization and clean code serve as the foundation.

1. E.g., Google PageSpeed: https://developers.google.com/speed/pagespeed/
2. E.g., Google Lighthouse: https://developers.google.com/web/tools/lighthouse
3. E.g., Yahoo! YSlow: http://yslow.org/user-guide/

After completing this chapter, you will have a better understanding of how it's really "all about WordPress", and how we can work *with* the software to create themes that are secure, well-tested, and well-optimized. Doing so enables us to build high-quality themes that are appreciated by both human visitors and search engines.

Building secure themes

On the Web, security is "mission critical" for success. Just as a chain is only as strong as its weakest link, so too is WordPress only secure as its weakest plugin or theme. For example, if you are running a default installation of WordPress with only the default plugins and themes, your site would be as secure as WordPress itself, which is considered very secure. Each plugin that you add, however, may or may not be as secure as WordPress. Likewise with your theme, if it is less secure than WordPress, then the entire site is only as secure as the theme.

Fortunately, securing a WordPress theme is generally much less work than securing an entire website. Even better, themes built with the WP API generally are secure out of the box, with no further security measures required. For example, when using high-level functions like `get_bloginfo()` and `wp_list_categories()`, database queries are sanitized by WordPress' internal functionality, so there's no need to take extra steps to sanitize the data.

As you develop a theme, there are a few things to keep in mind to help ensure an optimal level of security:

- Trust WordPress – use the WP API whenever possible.
- Trust nothing else – all data entered by the user or retrieved from the database should be considered unsafe.
- Validate all input data, sanitize all output data.

To put these points in perspective, let's return for a moment to the general theme built in the previous chapter. If you revisit each theme template, you will see that they are built with the WP API. All of the data is output via WP's own template tags, and there are no weird custom functions that might introduce any security vulnerabilities. Such themes may be considered to be as secure as WordPress itself.

Now let's say that we wanted to add a front-end form that enables visitors to submit their own posts. Adding such a feature would introduce a new vulnerability that may be exploited if not properly secured. Thus, in order to keep things secure, we apply the rule, "validate all input data, sanitize all output data". On the front-end, this means that we want to validate all data that the user enters into the form. This can be done either client-side (using JavaScript) or server-side (using PHP). As we'll see later in this chapter, WordPress provides plenty of template tags to use when validating user input.

Contrary to what one might expect, after the validated data is entered into the database, it still must be treated as unsafe. This is because of the dynamic way in which database content is used. For example, it is fine for post titles to contain single or double quotes when displayed inside of heading tags (e.g., `<h1>`, `<h2>`), but not when displayed inside of tag attributes. To understand why, consider the following code:

Tips to Keep WordPress Secure: https://m0n.co/securewp

```
<a href="<?php the_permalink(); ?>" title="Read the full
article: <?php the_title(); ?>"><?php the_title(); ?></a>
```

When used in the Loop, this code will display a link that uses `the_title()` for both the link text and title attribute. If none of the post titles include any quotes, this code is fine. If, however, some of the post titles include quotes, using `the_title()` for the value of the title attribute can lead to broken markup. For example, if our post title was something like this:

```
This post title uses "double quotes".
```

The double quotes would "break" the markup used to display the title. Viewing the source code on the front-end, this would be displayed as:

```
<a href="https://example.com/this-post-uses-double-quotes/"
title="This post title uses "double quotes"">This post title
uses "double quotes"</a>
```

Notice the problem with this code? The double quotes included in the value of the `title` attribute can cause browsers to misinterpret the syntax and break things when displaying the web page. Thus, the title data may be fine in one context (the link text), but not in others (the `title` attribute).

Likewise with other data — it may have been properly validated for one purpose, but may be used for other things as well. This is why it's necessary to sanitize all output data, even if it's validated before it gets to the database.

So it all depends on context. Without knowing the context in which the data was collected, there is no way of knowing whether or not it is safe to use. Incidentally, WordPress provides its own tag for displaying the title attribute:

```
<?php the_title_attribute(); ?>
```

This tag sanitizes the output data to make it safe for use in attributes. In this case, the sanitization includes converting quotes to their HTML equivalents, for example:

```
This post title uses &ldquo;double quotes&rdquo;
```

With the quotes properly converted to their respective entities, the title output by `the_title_attribute()` is safe for use in attributes.

This example illustrates why it is important to always validate input data and sanitize output data. To make developers' lives easier, WordPress provides a complete set of template tags that may be used to validate and sanitize any data that's required.

Validating data

Validating input and sanitizing output are important concepts, so let's take some time to study each in depth, and then look at some useful WordPress functions that make our development lives easier.

Whenever accepting input data, it is necessary to validate it. Data validation is all about making sure that the data is exactly what it is


Front-End Techniques

8
Front-End Techniques

Ride the wave

The field of web development evolves rapidly, with new tools, techniques and technologies constantly opening new doors and new ways of building websites. Over the years, web design and development have branched into numerous disciplines.

If you're looking for work these days, you'll need to know whether you are a UI Designer, UX Designer, Frontend Developer, Interaction Designer, SEO Specialist, Design Coordinator, or Full-Stack Developer. Within each of these areas, the methods and tools also are diversifying and evolving rapidly. It's almost a full-time job just keeping up with everything.

To put things in perspective, when I first started in web development over 10 years ago, we used CSS. Now, in addition to CSS, we have LESS, SASS, OOCSS, SMACCS, BEM, and so on[1–4]. The list of acronyms grows every year, along with the amount of awesome stuff that you can do with all of the new technology. The good news is that plain old CSS still works great, but you are encouraged to branch out and explore as many tools as possible; each brings something new to the table and will improve your own understanding and experience.

In this chapter, we stay focused on theme building and explore some useful techniques for customizing the front end of your theme. These

1. BEM approach to front-end development: https://en.bem.info/
2. SMACCS (Scalable and Modular Architecture for CSS): http://smacss.com/
3. Dive in (top results for OOCSS): https://www.google.com/search?q=oocss
4. Job Titles in the Web Industry: https://wp-tao.com/732

techniques draw upon what we've learned so far in the book, and will help to get you started in the dynamic field of front-end development. Note that all of the examples for this chapters are included with the book as demo files, so you can follow along and see how each technique works. So surf's up, let's hit some waves!

Design tips

If you're new to front-end development, here are some useful things to keep in mind as you learn the ropes. These are just general tips for those who may be unfamiliar with some of the basics.

Inspect your code

Just a reminder that cool browsers like Chrome and Firefox include built-in code inspectors that make it easy to check your source code while viewing it in the browser.

In Chrome, you can right-click on any element in a web page, and then select "Inspect Element". That will bring up the code inspector, which you should get to know – it is a huge timesaver.

In Firefox, you can right-click on any element and select either "Inspect Element" or "Inspect Element with Firebug" to use Firefox's built-in code inspector or Firebug extension, respectively.

Other browsers may have similar tools, check 'em out.

✳ Vertical space

Web pages are not like pieces of paper — there is no fixed length that requires designers to cram everything to fit on the page. This gives designers and developers the freedom to use plenty of vertical space when writing their CSS. Here are some good ways to ensure that things don't look all mashed together:

- Define a healthy line-height for text
- Apply ample vertical margins to block-level elements

Making sure that page items have plenty of room to breathe can make a lot of difference in the overall look and feel of a theme.

✳ Fonts

Thanks to the ability to define custom fonts in our stylesheets, it's easy to get carried away and use too many. When selecting fonts for your design, a good rule of thumb is to keep it simple. Using a minimal number of fonts keeps your design looking clean and organized, and also improves the load time of your pages.

Further, use fonts that are easy to read. When defining font styles, use sufficiently large font-sizes with ample line-height.

✳ Colors

Unless the theme design calls for it, it's best not to go crazy with colors. When getting into CSS, it's easy to get carried away and define a different color for everything, but doing so generally results in

cluttered, busy looking designs. Instead, use a strict color scheme that includes a limited number of colors.

Further, it's a good idea to use well-contrasted colors for text and background elements. For example, black or dark grey text on a white or light background is always a safe choice, and will make it easy for visitors to read and absorb the site's content.

✳ Frills

As you get into CSS, it's good to experiment with many different properties, but in general it's best to avoid adding a bunch of bells and whistles to your theme. For example, CSS makes it easy to apply borders, shadows, outlines, underlines, opacity, colors, and much more. This makes it easy to get carried away when styling your theme, but it's best to keep things simple and focus on the content. That is, the design should emphasize, not detract from, the site's content.

Rule of thumb: avoid bells and whistles and focus on content. Just because you *can* do something doesn't mean you *should*, and in general the old axiom "less is more" certainly applies.

Page layout

Let's say that we've just completed creating all of the template files for a theme and are now ready to add some styles with CSS. Before diving in and styling the details, take a moment to define the basic page layout. Some common layout configurations include:

- full-width header and footer with fixed-width content area
- full-width header and footer with fluid-width content area
- fixed-width header, footer, and content area
- fluid-width header, footer, and content area
- fixed-width layout with one sidebar
- fixed-width layout with left sidebar, content area, and right sidebar

And the list goes on and on. There are endless possible ways to organize page content, and infinitely more when `@media` queries are used for changing layout based on screen size. As we'll see in the next section, `@media` queries enable us to create responsive layouts that

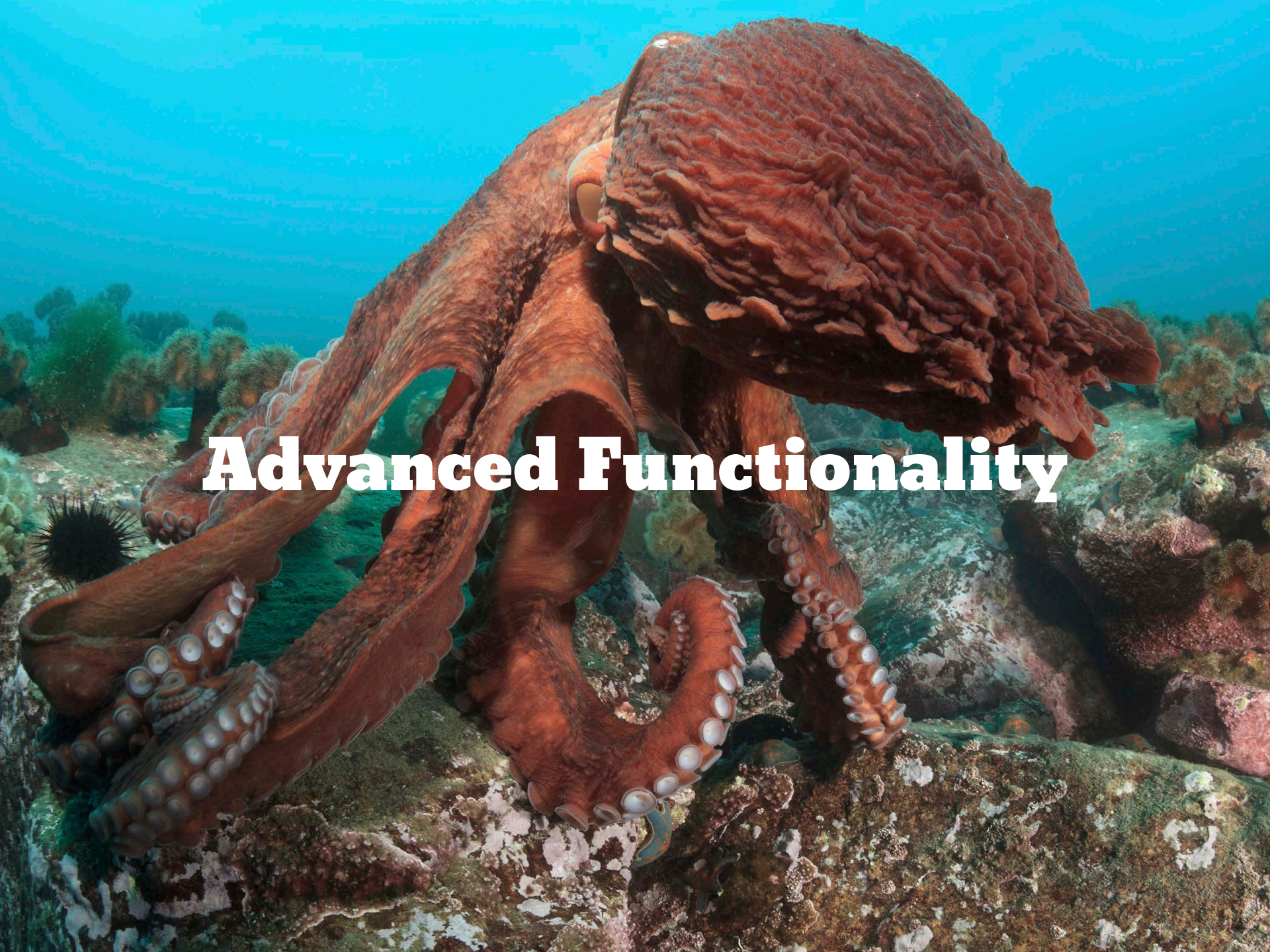
Multiple sidebars

WordPress makes it easy to define custom sidebar templates by simply naming them `sidebar-left.php`, `sidebar-right.php`, `sidebar-custom.php`, and so forth. Then to include a custom sidebar template, just add:

```
<?php get_sidebar('custom'); ?>
```

`get_sidebar()` accepts a `$name` parameter that corresponds to the second part of the filename. So the above tag will display the contents of `sidebar-custom.php`. To learn more, visit the WP Codex[1]:

1. https://developer.wordpress.org/reference/functions/get_sidebar/


Advanced Functionality

9
Advanced Functionality

Anything is possible

At this point in the book, we've covered many aspects of theme development, but we have yet to explore some of the advanced functionality that is made possible with the WordPress API and the functions template, `functions.php`. In this chapter, we dive in to some essential techniques for building some truly awesome themes.

Most of the techniques in this chapter are entirely plug-n-play, which makes them easy to implement in any WordPress theme. Each technique serves as a general example, and may be modified and extended to meet specific design requirements. Later in the book, we'll see how some of the techniques from this chapter are implemented in an actual theme[1].

As you go through the techniques in this chapter, keep in mind that it is entirely up to you to decide which functions and features to include or not include in your theme. If you're building a simple theme, then keep it simple and use only what is required. If you're going all out, then have fun and go all out. As long as you're using the WordPress API to make it happen, then "it's all good", as they say.

It also may be useful to keep a code library or some notes as you learn new techniques and implement them in your own themes. It can be a lot to absorb up-front, but as you continue in theme development, it all just "clicks", and you realize that, indeed, anything is possible.

1. See the 2020 Theme Walkthrough in Chapter 12.

Adding support for Theme Features

WordPress provides a number of built-in, ready-to-use features that can be added easily to any theme[1]. Theme Features include functionality such as custom backgrounds and headers, navigation menus, and post thumbnails. As with 99% of the techniques in this chapter, these techniques should be implemented via the theme's `functions.php` file. Let's go through each of WordPress' Theme Features and see how to register theme support.

✳ Custom Post Formats

Custom Post Formats, or more accurately, Post Formats, are a predefined set of formats that may be assigned to posts[2]. Post Formats make it possible to customize the appearance of different types of posts to reflect their content[3]. For example, aside posts may include only a link and description, while a video post may display only a video and title, and so on, for each supported Post Format.

As we've seen previously, support for Post Formats is registered with the tag, `add_theme_support()`[4]. For example, to enable the "aside" post format, we can add the following to `functions.php`:

```
add_theme_support('post-formats', array('aside'));
```

In this tag, the first parameter refers to the Theme Feature that we would like to support, which in this case is `post-formats`. The next parameter specifies the formats that we would like to support. To support all nine Post Formats, we would add this to `functions.php`:

```
add_theme_support('post-formats', array('aside', 'gallery',
'link', 'image', 'quote', 'status', 'video', 'audio',
'chat'));
```

These are the only nine formats allowed by WordPress. At the time of this writing the WP API does not provide a way to add new Post Formats.

Using Post Formats

Once Post Formats are supported, here are some useful template tags for using them in your theme[5]:

`get_post_format_link()` – Returns a permalink for a post-format archive. May be used anywhere if a Post Format is provided. Example:

```
<?php
	$format = 'aside';
	$format_link = get_post_format_link($format);
	echo $format_link;
?>
```

This will display the URL for the aside-format archive, for example:

```
https://example.com/type/aside/
```

1. https://codex.wordpress.org/Theme_Features
2. https://wordpress.org/support/article/post-formats/
3. https://dougal.gunters.org/blog/2010/12/10/smarter-post-formats/
4. https://developer.wordpress.org/reference/functions/add_theme_support/
5. https://wp-tao.com/745

`set_post_format()` – set the post format of a post. This can be used in the Loop or anywhere if a post ID can be provided. Example:

```
<?php set_post_format($post->ID, 'gallery'); ?>
```

This will set the Post Format for the current post to gallery.

`get_post_format()` – Returns the post format of a post. This will usually be called in the the loop, but can be used anywhere if a post ID is provided. Here is an example from Chapter 4 (see content.php):

```
<?php if (have_posts()) :
      while (have_posts()) : the_post();
            get_template_part('content', get_post_format());
      endwhile;
endif; ?>
```

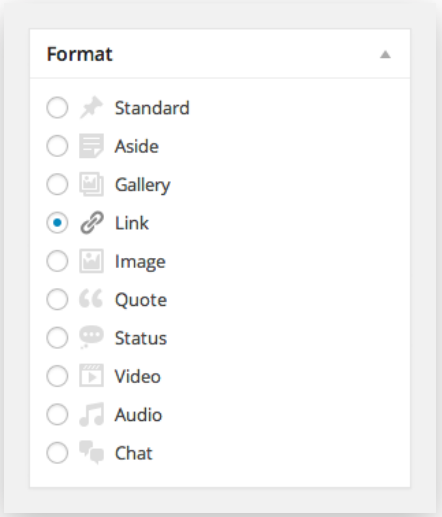
Here we are using `get_post_format()` as the second parameter in `get_template_part()`. This will include the correct template file for the current post format. For example, if post has an aside format, the included template file will be named `content-aside.php`.

`has_post_format()` – Returns a boolean (true or false) based on whether or not the specified Post Format is assigned to the post. This tag usually is called in the the loop, but can be used anywhere if a post ID is provided. Here is an example:

```
<?php if (has_post_format('aside'))
      echo 'This is an aside post.';
?>
```

Styling Post Formats

When theme support is added for Post Formats, WordPress displays the Format panel on the Edit Post page in the Admin Area. On the front-end, however, it's up to the theme to make use of Post Formats. Some themes take advantage of formats around every turn; other themes do nothing at all with formats. It's entirely up to the developer.



In addition to using template tags such as `get_post_format()` and `has_post_format()`, developers can use `body_class()` and `post_class()` to generate format-specific classes, for example:

- `.single-format-link` – via `body_class()` on single views
- `.tax-post_format` – via `body_class()` on archive views
- `.term-post-format-link` – via `body_class()` on archives
- `.format-link` – via `post_class()` on single and archive views

For each of these class names, "link" is the format. So in the theme's stylesheet, we could include styles for all supported formats like so:


Sharing & Selling Themes

10
Sharing & Selling Themes

Hello World!

Now that we've seen how to build our own themes, it's time to share our work with the rest of the world. There are many ways to share and sell your theme on the Web, and much responsibility that goes along with doing so. Yes the competition is fierce, but there is a great deal of potential, competition, and reward for anyone who is serious about giving it a go.

If you're developing themes for yourself, this chapter will show you how to make your theme better, using WordPress guidelines to improve the consistency, quality, and user-friendliness of your themes. If you're developing themes to share or sell, this chapter shows you how to ensure that your themes are up to snuff with all of the recommended features and functionality.

This chapter brings together most of the content covered so far in the book. We'll begin with an overview of how to prepare your themes for public release, and then go through the recommended guidelines in the form of a checklist that you can use to verify proper compliance. After going through all of the guidelines and recommendations, we'll look at some good places to share and sell your theme online, including the best of the best, your own website.

So get your themes ready and follow along — it's just a matter of time before you're promoting your themes and benefitting from them.

Preparing themes for public release

If you're new to the WordPress theme arena, you may be surprised at the number of "little things" that are required to produce a premium WordPress theme. For example, if you're preparing a new theme for the marketplace, you'll find that there is basically another layer of prep work that's required to meet all of the recommendations and guidelines. The good news is that, once you understand how and why these things should be done, you'll be able to integrate their implementation directly into your workflow.

In general, the degree to which your theme must comply with all of the requirements and recommendations is proportional to the number of people that will be using it. For example, if developing themes for your own use, it's probably fine to skip the internationalization stuff. Likewise, if you're developing a theme for a client, there is probably a clear set of goals that must be achieved; anything beyond those goals may or may not be necessary or covered in the budget. It's when you start releasing your themes for general consumption that you need to ensure that all the bases are covered. And if you're planning on submitting your theme to the WordPress Theme Directory, it is imperative that the required guidelines are followed to a "T".

1. https://codex.wordpress.org/Theme_Development
2. https://codex.wordpress.org/WordPress_Coding_Standards
3. https://codex.wordpress.org/WordPress_APIs
4. https://make.wordpress.org/core/handbook/best-practices/coding-standards/

Regardless of the target audience, however, all theme developers should strive to follow development standards, adhere to recommended guidelines, and practice thorough testing.

✳ Development Standards

As explained in the Codex, WordPress themes should be coded using the following standards[1]:

- Use well-structured, error-free PHP
- Use well-structured, valid HTML
- Use clean, valid CSS
- Use consistent, well-formatted JavaScript
- Follow design guidelines in Site Design and Layout

Adhering to these standards while writing code helps to avoid errors, improve readability, and simplify modification[2]. Additionally, it's important to reiterate that theme developers should use the WordPress API wherever possible[3].

To learn more about recommended best practices and coding standards, visit the Core Handbook, which is required reading for anyone who is working on the WordPress core[4]. Theme developers are encouraged to embrace these guidelines with the understanding that they are intended for core contributors. So while not technically required, it's a good way to keep your theme development aligned with the same standards used to build WordPress.

✳ Theme Guidelines

In addition to development standards, WordPress has a set of recommended guidelines for theme developers[1]. If you've been following along with the book, we've seen most, if not all, of the theme guidelines. To save you some time, the theme guidelines are summarized a bit later in this chapter as a set of two checklists that you can use to double-check that everything is too legit to quit:

- Theme Guidelines Checklist
- Template Files Checklist

It is important to note that the there are additional guidelines required for themes that are accepted at the WP Theme Directory[2–5]. For a great tool for checking that your theme is up to snuff with all of the guidelines, check out the Theme Check plugin, which automates everything and summarizes the results as actionable items for your consideration. Definitely a must-have for theme developers[6].

✳ Testing themes

Proper testing is a must. Taking the time to test your theme before it is released to the public provides numerous benefits. First, it generally improves the quality of the theme, which means a better user experience and happier theme users. It also means that you'll be addressing fewer help requests, which can add up quickly and gobble up precious time and resources. It's also about reputation: taking the time to test well can mean the difference between long-term success and failure in the WordPress-themes arena.

When it's time to test your theme, refer to Testing WordPress themes in Chapter 7 for a complete guide.

✳ It's all about quality

Always remember, *it's all about quality*. There is no shortage of half-witted theme developers out there who really couldn't care less about things like quality, integrity, and honesty. Sadly, rather than striving for excellence, some developers are more concerned about making money. This results in themes that may look great on the surface, but ultimately are found lacking upon closer examination. Such themes fail repeatedly and are not worth their weight in bytes.

So don't be like that. Strive for high-quality themes that are unique and awesome. That is what WordPress users have come to expect. Sure you can skip the sentimentality, take shortcuts, and spit out oodles of cheap, repetitive themes just to make a quick buck, but why take that route when so much more is possible simply by adhering to development standards and following guidelines. After doing so, you can be confident that your themes are ready for public release.

1. https://make.wordpress.org/themes/handbook/review/required/
2. WP Theme Directory: https://wordpress.org/themes/
3. WP Theme Review: https://make.wordpress.org/themes/handbook/review/
4. Theme Check Guidelines: https://make.wordpress.org/themes/handbook/guidelines/theme-check/
5. Theme Review Guidelines: https://developer.wordpress.org/themes/release/theme-review-guidelines/
6. Theme Check plugin: https://wordpress.org/plugins/theme-check/


Build a Starter Theme

11
Build a Starter Theme

Before we begin...

This chapter is the first of the book's two theme walkthroughs. This first walkthrough is meant to be simple, as a standalone guide to building your own starter theme. If you've read the book this far, excellent; if not, that's fine too. We'll refer to previous chapters for each step, so you can learn more or refresh your memory as needed.

In this walkthrough, we will build a basic starter theme. This will be another example of how to build a complete WordPress theme. In the next chapter, we'll level up and build another complete theme with a full plate of features and all the trimmings.

✳ About the DIY Theme

The end-product of this walkthrough will be a lightweight, flexible starter theme that you can use to jump-start the development process. Instead of having to assemble all of the essential theme templates, you can just rename your starter theme and dive in. The DIY Theme:

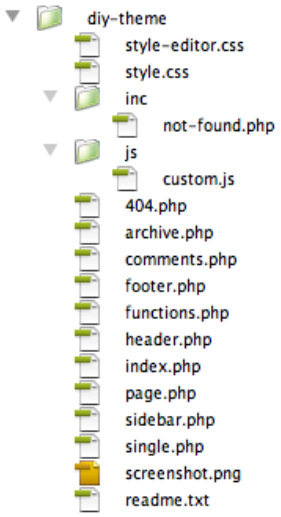
- Passes all theme recommendations
- Includes three nav menus and three widgetized areas (sidebars)
- Includes essential WordPress template tags
- Uses clean and valid HTML5 and CSS3
- Lightweight, flexible, and ready to use

The DIY Theme is included with this book. Grab a copy and follow along, or start from scratch and build it yourself.

Step 1: Set it up

Before touching any code, take a moment to set up your themes directory, subdirectories and files. First create the themes directory, named "diy theme" and add the following subdirectories and files:

```
/diy-theme/
	style-editor.css
	style.css
	/inc/
		not-found.php
	/js/
		custom.js
	404.php
	archive.php
	comments.php
	footer.php
	functions.php
	header.php
	index.php
	page.php
	sidebar.php
	single.php
```



These are common template files that generally are useful in any theme. By including them in our starter theme, we save ourselves from having to recreate all of these files every time we start a new theme. Note that we'll add the screenshot and `readme.txt` files after the theme is complete. Also, a reminder that the complete DIY Theme is included with the book. See Chapter 1 to learn more.

Step 2: Add the markup

Once the directory structure and blank template files are set up, it's time to add the primary markup, which for this theme is HTML5. After completing this step, our starter theme will generate pages that have the following base-level structure:

```
<!doctype html>
<html>
	<head></head>
	<body>
		<div class="container">
			<div class="box header"></div>
			<div class="box content"></div>
			<div class="sidebar"></div>
			<div class="box footer"></div>
		</div>
	</body>
</html>
```

This step is all about setting up a foundation of well-patterned code, which helps to minimize repetition and keep things consistent. Knowing in advance what our pages' markup should look like enables us to keep our theme template concise and consistent throughout. That, in turn, makes the theme easier to customize, extend, and style. Keeping the importance of consistent markup, let's continue building the starter theme by distributing this base markup to the appropriate template files.

✳ header.php

The header template, `header.php`, should contain the first six lines:

```
<!doctype html>
<html>
	<head></head>
	<body>
		<div class="container">
			<div class="box header"></div>
```

These lines will serve as the base markup for the header, including the `<!doctype>`, `<head>`, and opening `<body>` tags. Visit header.php in Chapter 4 to learn more about this template file.

✳ archive.php & index.php

Each of these templates contains a `.content` `<div>`:

```
<div class="box content"></div>
```

This `<div>` will contain all of the post content. To learn more, visit archive.php and index.php in Chapter 4.

✳ page.php & single.php

Each of these templates contains a `.content` `<div>`:

```
<div class="box content"></div>
```

This `<div>` will contain all of the post content. To learn more, visit page.php and single.php in Chapter 4.

✳ sidebar.php

Then, as expected, `sidebar.php` contains the sidebar `<div>`:

```
<div class="sidebar"></div>
```

Learn more about sidebar.php in Chapter 4.

✳ footer.php

Lastly, `footer.php` completes up the document structure:

```
			<div class="box footer"></div>
		</div>
	</body>
</html>
```

This completes the distribution of the theme's base markup. From here, we'll be moving on to other template files and adding base markup as needed. Learn more about footer.php in Chapter 4.

✳ comments.php

To `comments.php`, add the following markup:

```
<div id="comments" class="comments-wrap">
	<div class="comments-list"></div>
	<div id="respond" class="comments-respond"></div>
</div>
```

This will serve as the base markup for the comments area on Posts and Pages. Learn more about comments.php in Chapter 4.


2020 Theme Walkthrough

12
2020 Theme Walkthrough

2020 vision

So far in the book, we've seen how to build three basic themes: a simple theme, a general theme, and a starter theme[1]. In this chapter, we walk through the 2020 theme, which is a fully equipped *premium* theme. We'll look at 2020's features and functionality, discuss how it was built, and explore some key aspects of its design. At this point in the book, it's assumed that you have an understanding of theme development, so this chapter is less tutorial-based and more like a "guided tour" through the 2020 and its many awesome features.

About the 2020 theme

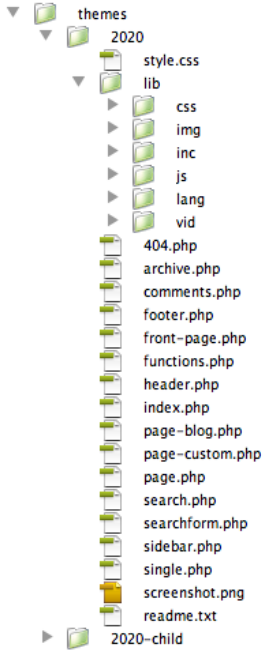
The 2020 theme is a premium theme that makes use of the techniques and principles explained in the book. On the front-end, 2020 is clean, bold, and beautiful. Under the hood, the template code is lightweight, consistent, and modular. It's good to go out of the box, and is ideal for learning, customizing, and even as a template for building your own premium themes. Here are just a few of the highlights of 2020[2]:

- Clean, secure, built with the WordPress API
- Satisfies all WordPress theme recommendations
- Built with valid HTML5 and CSS3
- Responsive, progressive, modular design
- Built with the shapeSpace starter theme

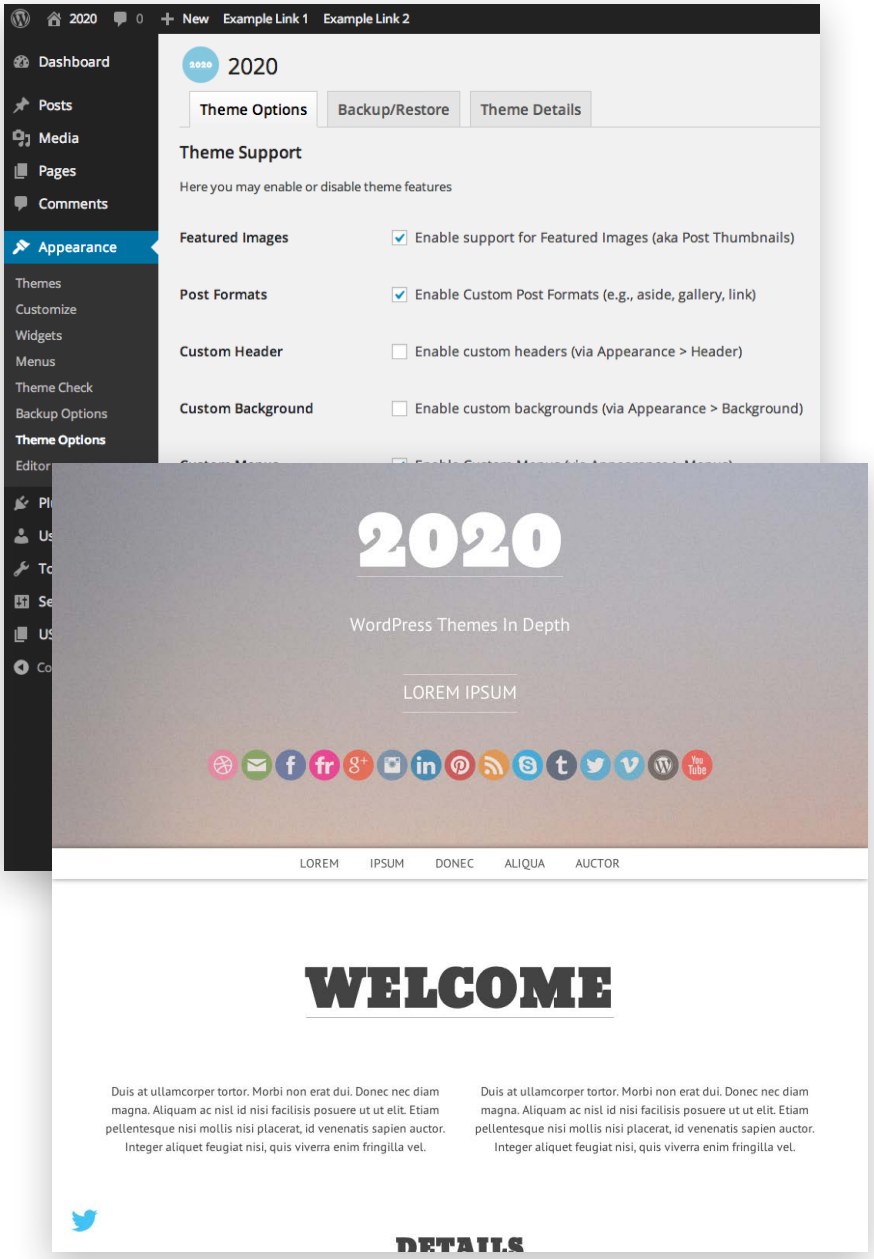
1. See Chapter 3, Chapter 6, and Chapter 11, respectively.
2. See the 2020 `readme.txt` for a complete list of features

And those are just the spotlight features, other features include:

- Unlimited Content Sliders
- Sticky Dropdown Menu
- Lightbox Functionality
- Theme Customizer
- Tabbed Theme Options Page
- 2020 Child Theme
- Contact Form
- Custom Fonts
- 404 Email Alerts
- Breadcrumb Navigation
- Full-size Background Videos
- 3 Custom Nav Menus
- 3 Widgetized Areas



2020 brings all of this and more with lean template files and minimal code. If you've downloaded 2020 and are wondering about the relatively large file size, it's mostly due to the deluxe set of bundled media files[1]. To say a few thousand more words about 2020, here are a few choice screenshots, showing the file structure (above), options page (upper right), and custom front-page template (lower right).



1. 2020 is equipped with 6 slides (jpg format), 6 lightbox images + thumbnails (jpg format), 10 featured images (jpg format), 28 full-size background images (jpg format), 3 sample videos each in 3 formats (mp4, ogv, webm) + screenshots, and a set of social-media icons.

2020 overview

The basic development process for the 2020 theme looks like this:

- Step 1 – add media files (icons, images, video, et al)
- Step 2 – add template code (PHP/HTML)
- Step 3 – style with CSS
- Step 4 – add JavaScript
- Step 5 – add a `readme.txt` and `screenshot.png`
- Step 6 – add a child theme
- Step 7 – test and refine

These steps are idealized for the sake of clarity. In reality, the steps tend to overlap and intersect. For example, wrangling template code seems to happen throughout most of development, and fiddling with CSS also can seem like an ongoing process. Even so, these steps are typical of theme development, so let's consider each of them.

Step 1: Media files

The 2020 theme includes a treasure-trove of open-source media files[1]. 2020 includes plenty of full-size images for backgrounds, sliders, lightbox examples, icons, and more. It also includes a set of videos that can be used for backgrounds, sliders, and elsewhere.

In general, it's a good idea to have all of your images, video, icons, and other media ready to go before adding any markup. As explained in Chapter 7, this includes things like optimizing the size of images and video[2]. Having a well-optimized set of media files ready and waiting makes it easy to include them as needed while developing the theme. Not having to interrupt your workflow to prepare a bunch of images saves time and keeps your attention focused on development.

Step 2: Template code

The template code (PHP and HTML) used for the 2020 theme is kept as lean and flexible as possible. For example, the markup used to structure the custom front-page template is basically a set of well-classed `<div>` containers. Containers can be added or removed as necessary to keep pages consistent and flexible. When simplified, the markup used in the 2020 theme looks like this:

```
<body>
	<div class="container fluid-width header">
		<div class="section"></div>
	</div>

	<div class="container fluid-width header-nav shdw1">
		<div class="section"></div>
	</div>

	<div class="container fixed-width clear padding-48">
		<div class="section intro"></div>
	</div>
```

continues ›

1. See the 2020 theme's `readme.txt` for complete documentation.
2. See Optimizing theme performance in Chapter 7.

Continue the journey...

https://wp-tao.com/wordpress-themes-book/