

A Beginner's  
Guide to  
WordPress

Theme  
Development

---

Ever wanted to get started with Theme Development in  
WordPress? This is your opportunity.

by Alex  
Denning, for  
[WPS shout.com](https://wps shout.com)



# A Beginner's Guide to WordPress Theme Development

## CONTENTS

- 3. CHAPTER 1: The fundamentals of any WordPress theme.
- 6. CHAPTER 2: The index.php and style.css files.
- 11. CHAPTER 3: The header, sidebar and footer.
- 14. CHAPTER 4: The single, comments and page files.
- 18. CHAPTER 5: The archive, home and functions files.



# A Beginner's Guide to WordPress Theme Development

# 1

In the chapter, we'll look at the very basics and fundamentals of any WordPress theme.

# WPSHOUT

## A Beginner's Guide to WordPress Theme Development

Welcome along to the first of WPShout's themed weeks! You're currently reading the first chapter of a free eBook from [WPShout.com](http://WPShout.com).

Getting started with WordPress theming can be a daunting prospect, and before we start, I'll say this now; I'm going to assume a good solid understanding of both CSS and HTML. Good places to gain this knowledge are [CSS-Tricks](#) and [Nettuts+](#). We are going to go quite quickly in order to rattle through this in a week. If you need any further explanation, then leave a comment and I'll be happy to help out.

A couple of things to sort out first - you'll need to get yourself a code editor. If you're using Windows, the free [Notepad++](#) is an excellent tool to have and if you're on a Mac then I'm sure there are plenty of great free editors, but the one that everyone raves about is called [Coda](#) (and it's not free). If you're serious about design then Coda will be worth it in the long term. The only other quick thing we need to do is install WordPress locally. Conveniently I posted how to do this just the other day; follow that tutorial for all the details.

### THE FUNDAMENTALS OF A WORDPRESS THEME

A WordPress theme is made up of a number of different files, and they all contain a separate section of the page; the header will contain the title and navigation, then the index will contain the main content area (or on a post, the single file does the job). The sidebar obviously contains the sidebar and the footer contains the footer and closes off the HTML. This all sounds very straightforward, but the important bit is how you can just have a single file, change it once and you will change your whole site. Change your footer and that change will be reflected sitewise, not just on a single page.

Expanding on this, a post page is made up of four files: the header.php file for the header, the single.php file for the post content, the sidebar.php file for the sidebar and the footer.php file for the footer. You can have the same header.php, sidebar.php and footer.php files for *the whole site*, and so when you make a change, this change comes immediately sitewise.



# A Beginner's Guide to WordPress Theme Development

Throughout this eBook we'll be looking at all the different the files that exist in WordPress, but before we do, it is very important you understand how it all fits together, which you should hopefully do now.

# WPSHOUT

## A Beginner's Guide to WordPress Theme Development

# 2

In this chapter we really get stuck in to development, looking at the `style.css` and `index.php` files.

# WPSHOUT

## A Beginner's Guide to WordPress Theme Development



**CLICK TO  
DOWNLOAD**

Today's download:  
the theme files for  
Day 2

**A**fter the excitement of learning about template files in the last chapter, it's time to move and look at the most important files of all WordPress themes: the `index.php` and `style.css` files.

First off, the `style.css`. This is our stylesheet. As I said in the previous chapter, this isn't a design series, so I'm not going to dwell on it too much, but there are some important parts of a stylesheet which WordPress needs that tell it some info about the theme. The theme we'll be creating this week is called Champion (download above). It's based on the Default WordPress theme for ease of use. Download the theme and unzip it. Open up the `style.css` file and you'll see something like this:

```
/*  
Theme Name: Champion  
  
URI: http://wps shout.com  
Description: description  
Author: Alex Denning  
Author URI: http://wps shout.com  
Version: 1.0  
*/
```

# WPSHOUT

## A Beginner's Guide to WordPress Theme Development

And that's all you need to make a stylesheet WordPress-ified. Moving on, the `index.php` file:

WordPress has something called a file hierarchy which means it will look for a file specific to the page first, but if it can't find it then it will use the `index.php` file. For example, posts have a hierarchy like so:

For posts, first, WordPress will look for the `single.php` file, but if it isn't found then it will use the `index.php` to display the post. [WPCandy](#) explains this really well with a great [image](#), but an example of our own:

For author archives, first WordPress will look for the `author.php` file, then the `archive.php` file and if it can't find that it'll use the `index.php` file to display the author archive. It's the same with all types of post, page or archive; **they all revert to the `index.php`**. This is why it is the most important file of them all.

So let's get started. Open up the theme files again and open the `index.php` file. We don't need any other files because as I've just pointed out, all types of page revert back to the `index.php` file so to prove the point, today it is the only file we're going to use. As the week goes on we'll be adding more files back in, don't worry! Open up the `index.php` file and look for line 49, which starts with:

```
<?php if (have_posts()) : ?> <?php while (have_posts()) : the_post(); ?>
```

This is the WordPress Loop, and the most important part of any WordPress theme. Today is exciting; we're using the most important template file and the most important part of any template file!

### THE LOOP AND TEMPLATE TAGS EXPLAINED

So what is this Loop? It's the thing that goes and fetches posts. With the loop started, look at the next couple of lines, ignoring the opening `<div>`s. They read:

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

These are your very first template tags. Template tags are pieces of PHP (which always start and end `<?php` and `?>`) which tell WordPress *for the current post, get*



# A Beginner's Guide to WordPress Theme Development

*this piece of information.* For example, the first template tag used here, the `_permalink`, tells WordPress *for the current post, get the post permalink*. Used inside an anchor and you've got yourself a link to current post. The next template tag, the `_title`, as you might have guessed, outputs the title of the post. Add that all together, inside an H2 tag and you've got yourself a title which when clicks goes to the post's post page.

The final template tag above is the `_content`. This tells WordPress *for the current post, go and find the contents of it and display it*.

The next part of the file reads (again, ignoring the divs):

```
<?php endwhile; ?>

<php next_posts_link(' Older Entries') ?>

<?php previous_posts_link('Newer Entries &raquo;') ?>

<?php else : ?>

<h2>Not Found</h2>

<p>Sorry, but you are looking for something that isn't here.</p>

<?php endif; ?>
```

The first part, *endwhile* tells WordPress *when you've finished displaying all the posts, display this*:. What is displayed are links to older and newer entries using the template tags `next_posts_link` and `previous_posts_link`.

The next part, *else* tells WordPress *if you can't find any posts, display this*:. You'll see that if no posts are found then a 'Not Found' displays. Finally, *endif* finishes the loop.

So there we have it; the most important part of any WordPress theme; the loop. What we've also done is get introduced to template tags. There are quite a few to get yourself familiar with, and you can find them listed on the [WordPress Codex](#).



## A Beginner's Guide to WordPress Theme Development

With the deciphering done, load up the theme to your WordPress install and activate it. You'll see that despite only having a stylesheet and a single `index.php` file, the theme loads fine. Click on a post and you'll see that gets displayed too. Pages, post archives, category archives and any page you like work fine too. Going back to what I said earlier – **all template file hierarchies fall back on the `index.php` file** – this proves it.

This also arises the question – “if I can do all this with a single file, why have more template files?” The answer is customisation. A change to the `index.php` file will be reflected throughout the entire site, so if you *just* want to change post pages then you wouldn't be (easily, see à) able to do this, which is why we have different template files (the pedantic might point out you could get round this with *if* statements, which is true, but it wouldn't be particularly practical at the end of the day).



## A Beginner's Guide to WordPress Theme Development

# 3

You've made it to the third chapter! Well done! In this chapter we're going to be looking at the header, sidebar and footer files.

# WPSHOUT

## A Beginner's Guide to WordPress Theme Development



**CLICK TO  
DOWNLOAD**

Today's download:  
the theme files for  
Day 3

Moving directly on from the previous chapter of the `index.php` and `style.css` files, today we're going to be creating three additional files: the header, sidebar and footer files. As you might have guessed, these are going to house the header, the sidebar and the footer.

### WHY HAVE SEPARATE FILES FOR THE HEADER, SIDEBAR AND FOOTER?

To answer the question, first download the latest copy of our theme and open up the `index.php` file. First thing that you should notice is that the header has been replaced with a piece of PHP - `<?php get_header(); ?>`. This is another of WordPress' template tags, and it's telling WordPress *go into the theme files, find the header.php file and display the contents here*. As your [theme](#) becomes more and more complicated, this allows you to create a single header and use it throughout your theme. At this point, you're probably opening up the `header.php` file. Good idea! Upon opening it, you'll see it's the same thing we had starting off our `index.php` file [yesterday](#). All that has changed is now it has a whole file to itself.

### DECIPHERING THE HEADER

# WPSHOUT

## A Beginner's Guide to WordPress Theme Development

Before we move on, a couple of important header points. You'll notice that the header doesn't display things like your content type, it uses template tags (I did say there were lots!) such as:

```
<?php bloginfo('html_type'); ?>
```

When this loads, what is displayed is your HTML type – have a look at the source code yourself – text/html gets shown. Template tags such as this one are used throughout the header to get the stylesheet url, title, pingback url etc etc. The reason for this is because every WordPress installation is different and so by using dynamic template tags, you can cater for all of these different installations at once.

### THE SIDEBAR

Look back in the index file and you'll see that the sidebar has gone too, and its place `<?php get_sidebar(); ?>`. Like the header, this tells WordPress *go into the theme files, find the sidebar.php file and display the contents here*. There's not much to decipher here; a couple of new template tags and only one completely new thing: widgets, which we'll discuss further on Friday as it requires the functions.php file.

### THE FOOTER

The final part of today's instalment is going to look at the footer. Like the header and sidebar, it has been removed from the index.php file and now resides in the footer.php file. Again, it is referenced with the `<?php get_footer(); ?>` function. Nothing much new here; again some more new template tags, but other than that it's much the same as the index.php file's footer we had [yesterday](#).

### WRAPPING UP

So there we are. We've split up our index.php file into a header, sidebar and footer. In the next chapter we'll be making use of these new files and creating the single.php file for posts.



## A Beginner's Guide to WordPress Theme Development

# 4

The penultimate chapter looks at the `single.php`, `comments.php` and `page.php` templates, for creating posts, pages and comments.

# WPSHOUT

## A Beginner's Guide to WordPress Theme Development



**CLICK TO  
DOWNLOAD**

Today's download:  
the theme files for  
Day 4

Today, continuing WPShout's series, "A Beginner's Guide To WordPress Theme Development", we'll be developing a file that handles posts – the `single.php` file. Download the latest copy of the theme we've been developing all week, 'Champion' with the link above, unzip it and you should notice a couple of new files have appeared! The two new files we need for today's instalment are the `single.php` file and the `comments.php` file. Let's first look at the `single.php` file.

### DEVELOPING THE SINGLE.PHP (POST PAGE) FILE

Upon opening the `single.php` file, it should look pretty familiar; the first line is `<?php get_header(); ?>` which, as we learnt yesterday, tells WordPress *find the header.php file and display the contents here*. Skip a line and you'll see:

```
<?php if (have_posts()) : while (have_posts()) : the_post(); ?>
```

That too should look familiar; it's the loop! Scroll down further and you'll see a couple of template tags - `<?php the_title(); ?>` and `<?php the_content(); ?>`. After that, scroll down to line 35 and you'll meet a new template tag: `<?php comments_template(); ?>`. Just like `get_header`, `get_sidebar` and `get_footer`, this tells WordPress *go and find the comments.php file and display the contents here*. You can probably guess where this is going – get the `comments.php` file open.

# WP SHOUT

## A Beginner's Guide to WordPress Theme Development

### GETTING TO GRIPS WITH WORDPRESS' COMMENTS

Open it up and... eek! It's complicated. The comments file is notorious for being the thing that must be done but no-one really likes doing. But that isn't going to stop us, is it? No. Let's get going. The first part of the code just checks to see if the comments file has been loaded directly, and if it has an error message gets displayed. Next, it checks to see if the current post is password protected, and if it is then the user is asked to enter a password to see the page. Leave this bit of code alone; it's important it is kept and, well, what has it ever done to you?

Next is the bit we're interested in: the comments loop. It starts by checking if comments are open:

```
<?php if ( have_comments() ) : ?>
```

And if they are then some text is displayed, showing the number of comments a post has. Skip a couple of lines to line 28 where an ordered list is opened and inside that ordered list is `<?php wp_list_comments(); ?>`. This is another of WordPress' template tags, and it spits out all of a post's comments. There are other ways of displaying comments (that offer more customisation), but that is a post for another day; as I said, it's quite a complicated subject.

Gloss over the next part (which is pretty self explanatory) to line 49. Here starts the form that you see on most blogs – this is the part where you fill out your name, email, website and comment. There isn't really much need to customise this or change it in any way, so we won't. Instead, we'll go back to the `single.php` file and finish up with that.

### FINISHING OFF WITH THE SINGLE POSTS

Open up the `single.php` file again and scroll down to line 37, just after where we left off. Here is something that again should look fairly familiar; it's the loop finishing off and saying *if no posts are found then display this:*. The `<?php endif(); ?>` closes off the loop and then we get to the familiar looking `get_sidebar` and `get_footer`, which we learnt about in the previous installment of the series.



# A Beginner's Guide to WordPress Theme Development

## CREATING A POST PAGE

This tutorial has focused heavily on posts, without a mention of pages. The good news is that pages run identically to posts and so to create a post page, all you have to do is save another copy of your `single.php` file as `page.php`. And you're done. I did say it was simple!

## WRAPPING UP

So there we are until next time. To recap this chapter's events, we've created a `single.php` page that handles posts, created a `comments.php` page that handles comments and a `pages.php` file that handles pages.



## A Beginner's Guide to WordPress Theme Development

# 5

The final chapter! In this chapter we're putting the finished touches to our theme with some more advanced template files.

# WPSHOUT

## A Beginner's Guide to WordPress Theme Development



Today's download:  
the theme files for  
Day 5

We're nearly there. This is the chapter of 'A Beginner's Guide to WordPress Theme Development'. We've got three files to tackle today, but there isn't too much I'm going to say on each of them. Let's start with the archive.php file. Download the latest copy of our theme, Champion, unzip it and load up the archive.php file in your text editor.

### CREATING AN AWESOME ARCHIVE PAGE IN WORDPRESS

According to the WordPress file hierarchy (which we discussed on day two), all archives – date, category, author and tag each have their own template file, but they also all fall back on a single file – the archive.php file. The idea here is that if you're clever, you can save yourself creating a couple of additional files. Upon opening the file, you'll be greeted with the familiar get\_header and the loop. Then, on line 14 starts a whole load of PHP *if* statements – if this is a category archive, display this, if this is a tag archive, display this etc etc. After that, on line 37 the loop swings into action and we have the familiar template tags we learnt on day two. Finally, on line 56 the standard "no posts were found" gets replaced with another *if* statement, changing it to "no posts were found under this category/tag etc etc". After that, the familiar get\_sidebar and get\_footer and the archive page ends. It's pretty similar to the index.php page, all that is happening is that there are a couple of *if* statements to change the titles according to the archive.

### CREATING AN (EQUALLY AWESOME) HOMEPAGE

# WPSHOUT

## A Beginner's Guide to WordPress Theme Development

Something you might have noticed is that so far we haven't created a specific homepage. Whilst there is a homepage, that's just the contents of the `index.php` file. At this point, it'd be appropriate to introduce a new template file: the `home.php` file. It's highest in the hierarchy for the homepage, so WordPress first looks for the `home.php` file and if that doesn't exist then it uses the `index.php`, which is why so far we've so far been creating a homepage with just the `index.php`.

### WHY CAN'T I JUST USE THE INDEX.PHP FILE?

Good question. You can't because the `index` file is at the bottom of all template hierarchies – if you don't have a specific template file for a certain type of page then WordPress displays the page using the `index.php` file. For that reason it is best to leave the `index` file as it is and create an additional page, `home.php` for generating a homepage. It's also one of those useful little tricks that are good to know as it allows you to stop using WordPress as a blogging platform and start using it as a CMS.

### DEVELOPING THE HOMEPAGE

In our example, we're not going to do anything with our `home.php` file, other than make it and copy and paste the contents of the `index.php` file into it, but as someone who is *getting started with WordPress theme development*, it is something that you should know exists and you've always got the option of customising the homepage further yourself.

### THE FUNCTIONS.PHP FILE

This is probably *the* most powerful template file there is. Effectively, it lets you run plugins from within your theme. It's not a page that gets displayed, it's a file that lets you run functions that you can display in other files. Common uses include:

- Widgets! Whilst you put where you want widgets to display in the theme files, but they're powered from the `functions.php` file.
- Theme options. Theme options pages are too created from the `functions.php` file. WPShout has a whole tutorial written on the topic here.
- Language – the `functions.php` file lets you set up the option for multi-lingual theming.



# A Beginner's Guide to WordPress Theme Development

As the functions file is just so complex, I'll just cover some basics. The code below will create a widget area 'Widget area 1' with an opening div before (which closes after) and the widget title gets an H3 tag:

```
<?php if ( function_exists('register_sidebar') )
register_sidebar(array(
'name' => ' Widget area 1',
'before_widget' => '<div class="widget">',
'after_widget' => '</div>',
'before_title' => '<h3>',
'after_title' => '</h3>',
)); ?>
```

To insert the widget in your sidebar you'll need to add the following to your sidebar.php file (or wherever you want to widgetise):

```
<?php if ( !function_exists('dynamic_sidebar') || !dynamic_sidebar('Widget
area 1') ) : ?>

<p>This area is widgetised! To make use of this area, put some widgets in
the 'Widget area 1' section.</p>

<?php endif; ?>
```

This is just the start though; take a read of [this](#) to find out how to build your own [theme options page in WordPress](#).

I hope this series has given you a good start with developing WordPress themes, and if it has make sure you [subscribe to WPShout by RSS](#) to hone your skills further. As I hope I've proved this week, it needn't be hard to become aware of what all the bits of code on your theme do, and you never know, perhaps in a year or two I could be reading your blog about advanced uses of WordPress!