Theme Basics

A theme consists of a stylesheet file called style.css. This file holds your basic style, the theme name, and data. Along with it are a bunch of PHP files, some absolutely necessary and some just good practice to make life easier on you or make interesting stuff happen.

These PHP files are called template files. You'll find index.php, which will be the main file for listings and search results, and is the fallback file for situations where there is no other template file available. Other common ones include sidebar.php, which holds the sidebar content, comments.php for comment functionality, and header.php/footer.php that are for your site's header and footer, respectively.

You may also have a single.php for single post view, and a page.php for static WordPress pages, and maybe a dedicated template file for search results (search.php), along with your category listings in category.php, and so on.

Add any number of page templates that you can apply to WordPress pages, and you get a tiny little glimpse of how versatile WordPress is.

With your template files, and WordPress functions as well as plugins and traditional PHP code, you can make your site behave in just about any way imaginable.

Maybe you want a specific product page to look completely different? Then create a page template and style it any way you like.

The best way to learn about themes is to use them. Install a theme on a test blog, play around, and then take a look at the files it consists of and start making changes.

Note: If you are taking this class, you will need to be already familiar with all of the basic administrative functions of WordPress: setting up Pages, Menus, Widgets, Plugins and Themes as well as the directory structure (for uploading files manually via a FTP program).

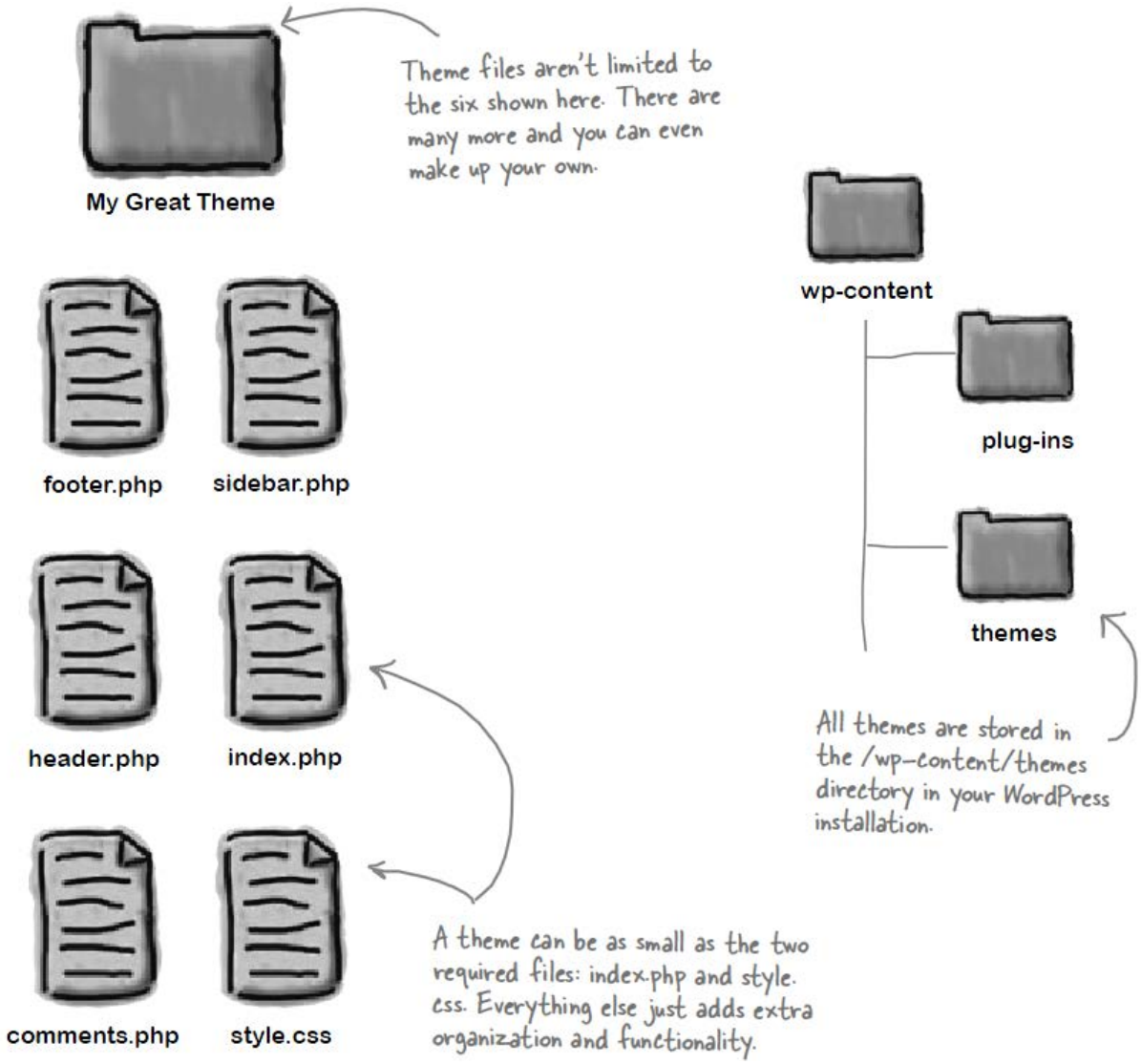


Figure 1 – WordPress theme files strucutre overview

A Few of the Main Template Files

404.php	Error page, served up when someone goes to a URL on your site that doesn't exist
archive.php	Page that displays posts in one particular day, month, year, category, tag, or author
comments.php	This file delivers all the comments, pingbacks, trackbacks, and the comment form when called
footer.php	Included at the bottom of every page. Closes off all sections. Copyright, analytics, etc)
functions.php	File to include special behavior for your theme.
header.php	Included at the top of every page. (DOCTYPE, head section, navigation, etc)
index.php	This is typically the "homepage" of your blog, but also the default should any other views be missing.
page.php	Template for Pages, the WordPress version of static-style/non-blog content
front-page.php	Template that will be recognized to be executed on home page, instead of index.php
home.php	Only used if you need to set a static home page and are displaying blog posts on a different Page, this will be used.
search.php	The search results page template
sidebar.php	Included on pages where/when/if you want a sidebar
single.php	This file is for displaying a single Post in full (the Posts permalink), typically with comments.
style.css	The styling information for your theme, required for your theme to work, even if you don't use. Of course you can have multiple .css files, but you need the style.css to start.
screenshot.png	This is the image thumbnail of your theme, for help distinguishing it in the Appearance picker.

Note: Starting point for themes and templates: http://codex.wordpress.org/Templates

Template Hierarchy

How does WordPress determine which template file to use?"

If your theme doesn't have an archive.php file, does WordPress just display a blank page? Absolutely not, it moves down its hierarchy of template files to find the next most appropriate file to use.

Ultimately, all paths in the WordPress templating world end at the **index.php** file.

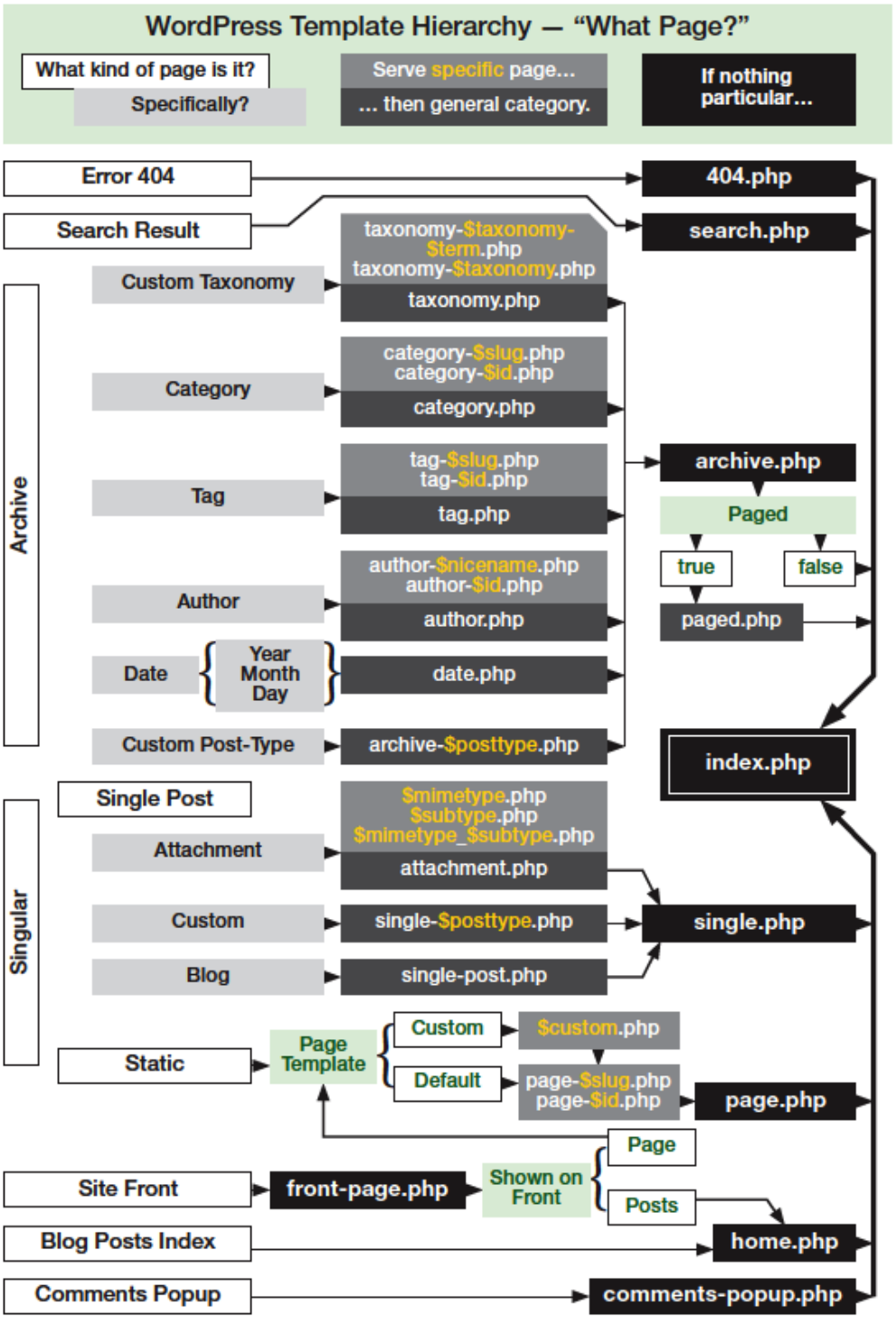


Figure 2 – WordPress template hierarchy

Note: http://codex.wordpress.org/Template_Hierarchy

Using the Template Tags

Although WordPress is written in PHP, it is in fact a framework in itself. You can use PHP to do stuff with your WordPress theme or plugin, but most of the functionality is managed with template tags.

If you open a theme file you'll find a lot of PHP-like functions, such as this one, for example:

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

That is a template tag, and it outputs the blog's name. Th e PHP part, which consists of <?php at first, and ; ?> at the end, tells WordPress to process whatever's inside it, and in this case it is the template tag bloginfo(). Inside the parenthesis you'll find the parameter, passed inside the quotation marks. In other words, 'name' is the parameter above.

You'll be using bloginfo() a lot in your themes, for example for finding the theme's directory.

Let's output an image called home.gif in a theme file:

```
<img src="<?php bloginfo('template_directory'); ?>/home.gif" />
```

You'll recognize the img HTML tag of course. The bloginfo() template tag has another parameter here, template_directory. This outputs the path to the theme's folder, called template directory instead of theme directory just to make things a little more complicated.

And then you just add the home.gif file name to complete the path, and you've got a potentially working image path in your theme. Of course, you would need the image in the theme folder as well. So template tags are basically PHP functions that can handle parameters to do different things. Some have default values, others don't, and some have more settings for you to play with than others.

Most of them will work anywhere in your WordPress theme files, but some need to be within the loop.

Note: A complete listing of template tags: http://codex.wordpress.org/Template_Tags/

The Include Tags

There are a couple of template tags that you'll find in just about any theme template file.

The include tags are basically PHP include functions to output the content of the necessary files within your theme. In other words, it is just a way to make it a bit easier to grab that header, footer, and sidebar.

```
<?php get_header(); ?>
<?php get_footer(); ?>
<?php get_sidebar(); ?>
```

You'll find them in your theme's index.php, for instance, and they automatically include header.php, footer.php, and sidebar.php, respectively, where the tags are placed. The first two won't take any parameters, but get_sidebar() actually supports alternate sidebars by adding a string to the tag, like this:

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

This would include sidebar-left.php rather than the default sidebar.php, so you'd need to create that file of course.

Should you want to include other files such as an alternate header, for example, you can use this nifty little PHP snippet:

```
<?php include (TEMPLATEPATH . '/altheader.php'); ?>
```

That is a traditional PHP include with TEMPLATEPATH that shows PHP where to look for the file, which is your theme folder. This is what the other include template tags do as well: they look in the theme folder for their respective files.

Conditional Tags

Conditional tags are very handy. You use them in your theme's template files, and as the name implies they are for setting various conditions. In other words, you can use them to display different things depending on the situation.

A very good example is the conditional tag is_home(), which checks if the page you're on is the homepage. Use it to output a greeting, since that is the polite
thing to do:

```
<?php if (is_home())
{
     echo '<p class ="welcome">Welcome to the site! </p>';
} ?>
```

This would output a paragraph with the class welcome, and the text within. So looking closer at this, you'll see it is a simple if clause asking if the page is_home(), and then an echo with the paragraph.

Very straightforward, so let's try something else. Say you've got a specific category that should have a different sidebar than the rest of the site, for example.

You can check that with the conditional tag is_category(), and then output another sidebar. Whenever it is another page within the site, use the traditional get_sidebar() include tag.

This code will replace the get_sidebar() PHP snippet in the theme's template files wherever it matters, which probably means files like index.php, category.php, single.php, and so on.

```
<?php if (is_category('very-special')) {

    get_sidebar('special');

} else {
    get_sidebar();

} ?>
```

Note: A complete listing of conditonal template tags:
http://codex.wordpress.org/Conditional_Tags

PHP Overview

WordPress is written in PHP, however you don't need to know a great deal simply to work with themes and even plugins. This is because much of the hard functionality like database access is handled by WordPress.

Nonetheless when editing a WordPress theme you will need to work with and around small snippets of PHP code.

So this is where I tell you I'm going to give you a 5-minute PHP crash course. ☺

Fortunately PHP is a simple language and you only need to understand a basic subset of how it works. Most of what you do in theme work is to call functions, place variables and very occasionally work in a loop or if statement.

1. What PHP tags look like?
   http://www.php.net/manual/en/language.basic-syntax.instruction-separation.php

   PHP is code wrapped in either <? … ?> or <?php … ?>. When the server sees these tags it knows to interpret all code in between as PHP and execute it on the server before serving up

the page to the end-user's browser.

2. What is a PHP comment?
   http://www.php.net/manual/en/language.basic-syntax.comments.php

   PHP has comments which are non-executed code. It is used solely for documenting what the following PHP code will do. It is wrapped in // to make a single-line comment or /* and */ to make a large comment block.

3. Using variables
   http://www.php.net/manual/en/language.variables.basics.php

   Variables are used for storing values, like text strings, numbers or arrays. When a variable is set it can be used over and over again in your script. All variables in PHP start with a $ sign symbol.

4. String variables
   http://www.php.net/manual/en/language.types.string.php

   String variables are used for values that contains character strings and have quotation marks " " around the value.

5. Conditional Statements
   http://www.php.net/manual/en/control-structures.if.php

   Very often when you write code, you want to perform different actions for different decisions. You can use conditional statements in your code to do this.

   if...else statement - use this statement if you want to execute a set of code when a condition is true and another if the condition is not true.

   elseif statement - is used with the if...else statement to execute a set of code if one of several condition are true.

6. How Loops Work
   http://www.php.net/manual/en/control-structures.while.php

   A loop tells the server to execute the same bit of code a set number of times usually with a changing variable. There are different types of loops (while, for and for each). The while loop will execute a block of code as long as a condition is true, otherwise it will skip over and continue with any other code.

7. Creating Functions

http://www.php.net/manual/en/functions.user-defined.php

A function is a block of code that can be executed whenever we need it. All functions start with the word "function()". A function has a name which can start with a letter or underscore (not a number).

Note: If yu want to learn more about PHP, you can register for the Introduction to PHP class at eClasses.org.

Creating Our Own Theme: HTML template

Before getting too in-depth with WordPress theme templating, we first need an HTML design that we can create a WordPress theme from.

There are various schools of thought on how to create a WordPress theme. Most either create a child theme from a popular existing theme or create a standalone one from scratch.

A quick review of the advantages and disadvantages:

Child Theming:

1. Saves alot of work.
2. Allows developer to stay up-to-date on new features of WordPress.
3. Requires a suitable parent theme.
4. Parent theme might bundle features not needed (commercial themes more likely).
5. Potentially a smaller degree of control.

Standalone theming (from scratch):

1. Can make markeup and design to exact specifications as per static design.
2. No unexpected functional changes (due from parent theme with or without upgrades).
3. No help getting started (though using a starting theme will help).
4. No help on keeping up to date on updates (if needed).

Note: For this class we will be creating **a theme from scratch**, not a child theme. We will use a "starter" theme to pick and choose existing template code to speed up development.

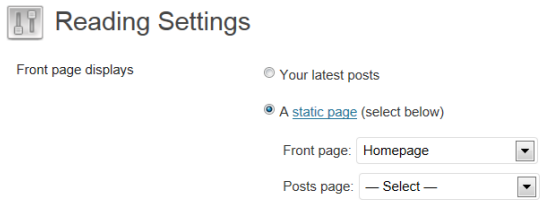
Figure 3 – Our HTML template we will re-create in a WordPress theme

Note: I included all the HTML files (.css and images) in this weeks lesson.

As you look over the template, it may not look like your typical blog website since it has various sections that do not focus on blog content, which is exactly the point.

Many WordPress websites do not have the blog content on the home page at all. Normally these web layouts have just the most recent post on front page and other content filling out the rest of the page such as images, video, static page content, and various links and feeds.

As I discussed in my Introduction to WordPress class, you can set your blog page to any Page that you create in the WordPress dashboard.



So you are not limited to only having your blog content on your home page. With our design, we will be using a little PHP to retrieve the latest blog post and display its summary on the front page with a link to the post which is on another page.

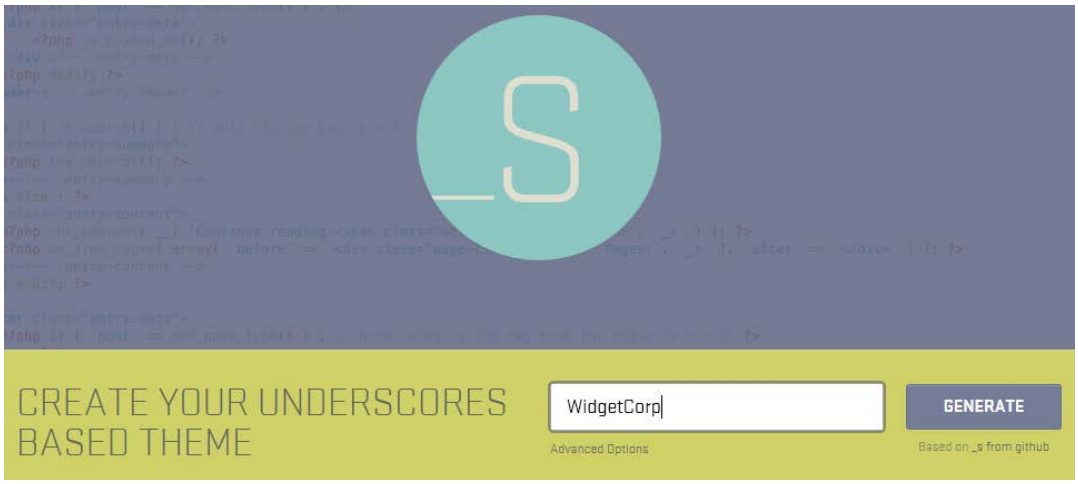
Using a "Blank" theme as a Guide

Even though we will be creating a theme from scratch, there is some template code that will be the same no matter what theme we create. So it makes sense for us to use a barebones theme and pick and choose the template code we will need in our own custom theme.

For seasoned WordPress developers, many will have their own starter theme that they use when creating new WordPress sites and just update as they need it for when new features of WordPress are added.

As I mentioned earlier, there are advantages and disadvantages of creating a theme from scratch and it will be up to you to decide what is best for your individual web projects.

The starting theme is called Underscores, which was created by the people of Automattic (creators of WordPress).



Note: Generate a starter theme by clicking the "Generate" button at http://underscores.me/

References:

1. General Theme Articles: http://codex.wordpress.org/Templates
2. Template Hierarchy: http://codex.wordpress.org/Template_Hierarchy
3. blog_info template tag : http://codex.wordpress.org/Function_Reference/bloginfo
4. Template tags: http://codex.wordpress.org/Template_Tags/
5. PHP include: http://php.net/manual/en/function.include.php
6. Understanding The WordPress Loop: http://codex.wordpress.org/The_Loop
7. Conditional tags: http://codex.wordpress.org/Function_Reference/is_home
8. get_sidebar template tag: http://codex.wordpress.org/Function_Reference/get_sidebar