



The Handbook

Codex

http://codex.wordpress.org/Theme_Development

Version Date

8 August 2005

Theme Development

The following article is about developing or designing your own WordPress Theme. If you wish to learn more about how to install and use Themes, review the documentation regarding [Using Themes](#).

You may wish to develop WordPress Themes for your own use or [for distribution](#). This topic differs from [Using Themes](#) because it discusses the technical aspects of writing code to build your own Themes rather than how to activate Themes or where to obtain new Themes.

Why WordPress Themes

WordPress Themes are files and styles that work together to create a presentation or look for a WordPress site. Each Theme may be different from each other, offering many choices for users to take advantage of in order to instantly change their website look. Why should you build your own WordPress Theme?

- To create your own unique WordPress site look
- To take advantage of [templates](#), [template tags](#), and [the WordPress Loop](#) to generate different web page results and looks.
- To provide alternative templates for specific site features, such as [category pages](#) and search result pages.
- To quickly switch between two site layouts, or to take advantage of a [Theme or style switcher](#) to allow users to change the look of your site.
- Design WordPress Theme(s) so that others may enjoy your designs through public release.

A WordPress Theme has many benefits, too.

- It separates the presentation styles and [template files](#) from the system files so the site will upgrade without drastic changes to the visual presentation of the site.
- It allows for customization of the presentation and web page results unique to that Theme.
- It allows for quick changes of the look and feel of a WordPress site.
- It takes away the need for a WordPress user to have to learn CSS, HTML, and PHP in order to have a good looking website.

Why should you build your own WordPress Theme? That's the real question.

- It's an opportunity to learn more about CSS, HTML/XHTML, and PHP.
- It's an opportunity to put your expertise with CSS, HTML/XHTML, and PHP to work.
- It's creative.
- It's fun (most of the time).
- If you [release it to the public](#), you can feel good that you shared and gave something back to the [WordPress Community](#) (okay, bragging rights!)

Anatomy of a Theme

WordPress Themes live in subdirectories residing in `wp-content/themes/`. This directory will hold all of the Theme's style sheet files, [template files](#), and images. For example, a Theme named "test" may reside in the directory `wp-content/themes/test/`.

WordPress includes two Themes in the download, a "Classic" and "Default" Theme. Each one is different and uses different functions and tags to generate their web page results and looks. Examine the files carefully for these Themes to get a better idea of how to build your own Theme files.

WordPress Themes consist of two main files, in addition to images. One is the style sheet called `style.css`, which controls the presentation (look) of the web pages, and the other files are the [template files](#) which control the way the web page generates the information from the Database to be displayed as a web page. Let's look at these individually.

Theme Style Sheet

The stylesheet, `style.css` **must** provide details about the Theme in the form of comments. **No two Themes are allowed to have the same details** listed in their comment headers, as this will lead to problems in the [Theme selection dialog](#). If you make your own Theme by copying an existing one, make sure you change this information first.

The following is an example of the first few lines of the stylesheet, called the style sheet header, for the Theme "Rose":

```
/*
Theme Name: Rose
Theme URI: the-theme's-homepage
Description: a-brief-description
Author: your-name
Author URI: your-URI
Template: use-this-to-define-a-parent-theme--optional
Version: a-number--optional
.
General comments/License Statement if any.
.
*/
```

The simplest Theme includes only a `style.css` file, plus images, if any. To create such a Theme, you must specify a set of templates to *inherit* for use with the Theme by editing the `Template:` line in the `style.css` header comments. For example, if you wanted the Theme "Rose" to inherit the templates from another Theme called "test", you would include `Template: test` in the comments at the beginning of Rose's `style.css`. Now "test" is the parent Theme for "Rose", which still consists only of a `style.css` file and the concomitant images, all located in the directory `wp-content/themes/Rose`. (*Note that specifying a parent Theme will inherit all of the template files from that Theme — meaning that any template files in the child Theme's directory will be ignored.*)

The comment header lines in `style.css` are required for WordPress to be able to identify a Theme and display it in the [Admin panel](#) > [Presentation](#) as an available Theme option along with any other installed Themes.

Note : *When defining the parent Theme, in the `Template:` section of the comment header, you must use the name of the directory of the style. For example, to use as parent*

template the Default Wordpress Theme, don't write `Template: WordPress Default`, but `Template: default`, because default is the directory of this Theme.

Theme Template Files

[Templates](#) are PHP source files used to generate the pages requested by visitors. Let's look at the various templates that can be defined as part of a Theme.

WordPress allows you to define separate templates for the various aspects of your weblog, however, it is not essential to have all these different template files for your blog to function fully. Templates are chosen and generated based upon the [Template Hierarchy](#), depending upon what templates are available in a particular Theme. As a Theme developer, you can choose the amount of customization you want to implement using templates. For example, as an extreme case, you can use only one template file, called `index.php` as the template for *all* pages generated and displayed by the weblog. A more common use is to have different template files generate different results, to allow maximum customization.

Basic Templates

At the very minimum, a WordPress Theme consists of two files:

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

The `index.php` [template file](#) is unique and flexible. It can be used to include all references to the header, sidebar, footer, content, categories, archives, search, error, and other web pages generated by the user on your site. Or it can be *subdivided* into modular template files, each one taking on part of the workload. If you do not provide any other template files, WordPress will use the built-in default files. For example, if you do not have either a `comments.php` or `comments-popup.php` template file, then WordPress will automatically use the `wp-comments.php` and `wp-comments-popup.php` template files using [Template Hierarchy](#). These default templates may not match your Theme very well, so you probably will want to provide your own.

If you want to subdivide the `index.php` master template file, as showcased in the Default WordPress Theme, you can. Such subdivision or modularization is optional. These modular template files may include these basic files:

- `header.php`
- `sidebar.php`
- `footer.php`
- `comments.php`
- `comments-popup.php`

These are the basic files needed to make a Theme. Place them with `style.css`, and `index.php` into their own subdirectory beneath `wp-content/themes/`, and you have a Theme.

Using these modular template files, you can put template tags within the `index.php` master file to include or *get* these units where you want them to appear in the final generated web page.

- To include the header, use the [get_header\(\)](#) template tag.
- To include the sidebar, use the [get_sidebar\(\)](#) template tag.
- To include the footer, use the [get_footer\(\)](#) template tag.

Here is an example of the *include* usage:

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

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

For more on how these various Templates work and how to generate different information within them, read the [Templates](#) documentation.

Query-based Templates

WordPress can load different [Templates](#) for different *query* types. It does this as part of the built-in [Template Hierarchy](#), or through the use of [Conditional Tags](#) within [The Loop](#) of a template file, like the `index.php` file.

Using the Template Hierarchy, if your Theme provides a template called `category.php` and a category is being queried, `category.php` will be loaded instead of `index.php`. If `category.php` is not present, `index.php` is used as usual.

If you wish to design a template for one specific category, so that only that category ID number will generate a different category page, and the rest will fall back to the `category.php` or `index.php` template file, name the file `category-#.php` where `"#"` is the category's ID number (found in [Manage > Categories](#) if you are logged in as the site administrator). If you set `category-6.php` to be the category page for Category 6, then it would display, overriding `category.php`. For a more detailed look at how this process works, see [Category Templates](#).

To generate a different template file based upon a [Conditional Tag usage](#) or query, if the conditions are met, then that template file will be used by WordPress, otherwise it will fall back on the next file in the [Template Hierarchy](#).

For example, to generate a distinctive style sheet in a post only found within a specific category, the code might look like this:

```
<?php
if (is_category(9)) {
    // looking for category 9 posts
    include(TEMPLATEPATH . '/single2.php');
} else {
    // put this on every other category post
    include(TEMPLATEPATH . '/single1.php');
}
?>
```

Or, using a query, it might look like this:

```
<?php
$post = $wp_query->post;
if ( in_category('9') ) {
    include(TEMPLATEPATH . '/single2.php');
} else {
    include(TEMPLATEPATH . '/single1.php');
}
?>
```

So if the condition or query is met, that the post is in Category 9, then it would display in the template file called `single2.php`. If the query is not met, then it would use the `single1.php` template file.

These query conditions are not limited to posts. You can also design different Templates for different [Pages](#). See that reference for details.

Theme Template Files List

Here is the list of Theme template files recognized by WordPress. Of course, your Theme can contain any other style sheets, images, or files. *Just keep in mind that the following have special meaning to WordPress.*

`style.css`

The main stylesheet. This **must** be included with your Theme.

`index.php`

The main template. If your Theme provides its own templates, `index.php` must be present.

`comments.php`

The comments template. If not present, `comments.php` from the "default" Theme is used.

`comments-popup.php`

The popup comments template. If not present, `comments-popup.php` from the "default" Theme is used.

`home.php`

The home page template.

`single.php`

The single post template. Used when a single post is queried. For this and all other query templates, `index.php` is used if the query template is not present.

`page.php`

The page template. Used when an individual [Page](#) is queried.

`category.php`

The [category template](#). Used when a category is queried.

`author.php`

The [author template](#). Used when an author is queried.

`date.php`

The date/time template. Used when a date or time is queried. Year, month, day, hour, minute, second.

`archive.php`

The archive template. Used when a category, author, or date is queried. Note that this template will be overridden by `category.php`, `author.php`, and `date.php` for their respective query types.

`search.php`

The search template. Used when a search is performed.

`404.php`

The [404 Not Found](#) template. Used when WordPress cannot find a post or page that matches the query.

These files have a special meaning with regard to WordPress because they are used as a replacement for `index.php`, when available, and when the corresponding [Conditional Tag](#) (a.k.a `is_*()` ; function) returns true. For example, if only a single post is being

displayed, the `is_single()` function returns 'true', and, if there is a `single.php` file in the active Theme, that template is used to generate the page.

Referencing Files From a Template

The WordPress Default Theme (based on Michael Heilemann's [Kubrick](http://binarybonsai.com/kubrick/) (<http://binarybonsai.com/kubrick/>) layout for WordPress 1.2) provides a good example of how queries are mapped onto templates.

The code `<?php bloginfo('template_directory'); ?>` inserts the URL of the template directory into the template output. You can append any additional URI information to this output to reference files in your Theme.

The code `<?php bloginfo('stylesheet_directory'); ?>` inserts the URL of the directory that contains the current Theme stylesheet into the template output. You can append any additional URI information to this output to reference files for your Theme, specifically those that are used by the stylesheet.

The constant `TEMPLATEPATH` is a reference to the absolute path to the template directory for the current Theme (without the `/` at the end).

Note that URIs that are used in the stylesheet are relative to the stylesheet, not the page that references the stylesheet. This obviates the need to include PHP code in the CSS file to specify directories. For example, if you include an `images/` directory in your Theme, you need only specify this relative directory in the CSS, like so:

```
h1 { background-image: URL(images/my_background.jpg); }
```

It is a good practice to use URIs in the manner described above to reference files from within a template, since, then your template will not depend on absolute paths.

Defining Custom Templates

It is possible to use the WordPress plugin system to define additional templates that are shown based on your own custom criteria. This advanced feature can be accomplished using the `template_redirect` [action hook](#). More information about creating plugins can be found in the [Plugin API](#) reference.

Plugin API Hooks

When developing themes, it's good to keep in mind the specialized group of [Plugin API](#) functions listed below. These plugin hooks are called as [Template Tags](#), either as a parameter to WordPress' `do_action()` function:

```
<?php do_action( 'hook_name' [, optional second parameter] ); ?>
```

or as functions in their own right (one exception noted below):

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

When you include the template-specific hooks in your theme, plugins will be able to run code or display data directly in it, without requiring their own template tag functions be

added. So it's important to include these plugin hooks in your templates especially if you plan on redistributing your theme, otherwise you risk breaking important features of a plugin which makes use of one or more of them. See [Current Hooks For Actions](#) for a full list of `do_action` hooks (note that most are not intended as template hooks for plugins).

Template Plugin Hooks:

`wp_head`

Goes in the [HTML](#) `<head>` element of a theme; `header.php` template. This hook has no second parameter. Example plugin use: add javascript code.

Usage: `<?php do_action('wp_head'); ?>` *-or-* `<?php wp_head(); ?>`

`wp_footer`

Goes in the "footer" of a theme; `footer.php` template. This hook has no second parameter. Example plugin use: insert PHP code that needs to run after everything else.

Usage: `<?php do_action('wp_footer'); ?>` *-or-* `<?php wp_footer(); ?>`

`wp_meta`

Typically goes in the `Meta` section of a theme's menu or sidebar; `sidebar.php` template. This hook has no second parameter. Example plugin use: include a rotating advert.

Usage: `<?php do_action('wp_meta'); ?>` *-or-* `<?php wp_meta(); ?>`

`comment_form`

Goes in `comments.php` and `comments-popup.php`, usually after the comment form. Its second parameter must be `$post->ID`. Example plugin use: display a comment preview.

Usage: `<?php do_action('comment_form', $post->ID); ?>`

For a real world usage example, you'll find these plugin hooks included in the default theme's templates.

Theme Development General Guidelines

Please be clear about the following in your documentation (a README file included with your Theme helps many users over any potential stumbling blocks):

1. Indicate precisely what your Theme and template files will achieve.
2. Indicate deficiencies in your Themes, if any.
3. Clearly reference any special modifications or notes to [comments](#) within the template and style sheet files and comment modifications, template sections, and CSS styles, especially those which cross template files.
4. If you have any special requirements, which may include custom RewriteRules, or the use of some additional, special templates, images or files, please explicitly state the steps of action a user should take to get your Theme working.
5. Try and test your Theme [across browsers](#) to catch at least a few of the [problems](#).

the users of the Theme may find later.

6. Provide contact information (web page or email), if possible, for support information and questions.

Since the concept of using Themes is currently new to WordPress, the additional effort you put in to make the Themes user-friendly will be greatly appreciated. Take time to read through [Designing Themes for Public Release](#), an article with good tips on preparing your Theme for the public. For criteria on how to release your Theme to the public, see [Promoting Your Theme](#).

References and Resources

To help you develop your WordPress Theme, here are some resources and references.

WordPress Resources

CSS and Layout

- [WordPress CSS Information and Resources](#)
- [FAQ - Layout](#)
- [Blog Design and Layout](#)
- [Developing a Colour Scheme](#)

Theme and Template Resources

- [Using Themes](#)
- [Template Files](#)
- [Creating Individual Pages](#)
- [Template Tags](#)
- [Stepping Into Template Tags](#)
- [The WordPress Loop](#)
- [The Loop in Action](#)
- [Designing Themes for Public Release](#)

Testing and Validating

- [CSS Fixing Browser Bugs](#)
- [CSS Troubleshooting](#)
- [Validating a Website](#)

General Resources

- [WordPress Lessons](#)
- [Know Your Sources](#)

References

- [Ryan Boren's Anatomy of a Theme](http://boren.nu/archives/2004/11/10/anatomy-of-a-wordpress-theme/) (<http://boren.nu/archives/2004/11/10/anatomy-of-a-wordpress-theme/>)
- [Moshu's Visual Anatomy of a WP v1.5 Theme](http://www.transycan.net/blogtest/2005/03/31/visual-anatomy-of-a-wp-15-theme/) (<http://www.transycan.net/blogtest/2005/03/31/visual-anatomy-of-a-wp-15-theme/>)
- [Templates and "is" functions, by Ryan](http://boren.nu/archives/2004/10/16/templates-and-the-is-functions/) (<http://boren.nu/archives/2004/10/16/templates-and-the-is-functions/>)
- [Chris J. Davis' Secrets of WP Theming, Part 1](http://www.chrisjdavis.org/2005/05/26/secrets-of-wp-theming-part-1/) (<http://www.chrisjdavis.org/2005/05/26/secrets-of-wp-theming-part-1/>)
- [Chris J. Davis' Secrets of WP Theming, Part 2](http://www.chrisjdavis.org/2005/06/02/secrets-pt-2/) (<http://www.chrisjdavis.org/2005/06/02/secrets-pt-2/>)
- [Chris J. Davis' Secrets of WP Theming, Part 3](http://www.chrisjdavis.org/2005/06/13/secrets-pt-3/) (<http://www.chrisjdavis.org/2005/06/13/secrets-pt-3/>)
- [Theme Cheat Sheet for WordPress 1.5](http://www.techwench.com/05-2005/15-theme-cheat-sheet) (<http://www.techwench.com/05-2005/15-theme-cheat-sheet>)
- [Dissection of a WordPress Theme](http://www.urbangiraffe.com/2005/04/12/themeguide1/) (<http://www.urbangiraffe.com/2005/04/12/themeguide1/>)
- [WordPress CSS Guides by Podz](http://www.tamba2.org.uk/wordpress/graphicalcss/) (<http://www.tamba2.org.uk/wordpress/graphicalcss/>)

Tools

[Plain Text Editors](#)

A list of plain text editors for all platforms.

[Developing a Colour Scheme](#)

Article on developing a colour scheme for your site.

[WordPress Index Builder](http://www.redalt.com/Tools/builder.php) (<http://www.redalt.com/Tools/builder.php>)

This tool will ask you a series of questions about your site to generate a working WordPress template page and CSS file. Criteria include number of columns and color matching.

[Flumpcakes CSS Optimizer](http://flumpcakes.co.uk/css/optimiser/) (<http://flumpcakes.co.uk/css/optimiser/>)

Optimize your CSS file.

Other Resources

- [Creative Commons Images](http://creativecommons.org/image/) (<http://creativecommons.org/image/>): Search for Creative Commons images that are free to share and use online.
- [Image * After](http://www.imageafter.com/) (<http://www.imageafter.com/>): A free stock image library for commercial and personal use.
- [stock.xchnng](http://www.sxc.hu/) (<http://www.sxc.hu/>): Photographs available free of charge to the public.