

## Theme Development

Languages: [English](#) • [日本語](#) • [한국어](#) • [Português do Brasil](#) • [Русский](#) • [ไทย](#) • [中文\(简体\)](#) • [中文\(繁體\)](#) • [\(Add your language\)](#)

This article is about developing WordPress Themes. If you wish to learn more about how to install and use Themes, review [Using Themes](#). 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 that work together to create the design and functionality of a WordPress site. Each Theme may be different, offering many choices for site owners to instantly change their website look.

You may wish to develop WordPress Themes for your own use, for a client project or to [submit to the WordPress Theme Directory](#). Why else should you build a WordPress Theme?

- To create a unique look for your WordPress site.
- To take advantage of [templates](#), [template tags](#), and [the WordPress Loop](#) to generate different website 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 site owners to change the look of your site.

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 site functionality unique to that Theme.
- It allows for quick changes of the visual design and layout of a WordPress site.
- It removes the need for a typical WordPress site owner to have to learn CSS, HTML, and PHP in order to have a great-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, and PHP.
- It's an opportunity to put your expertise with CSS, HTML, 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)

## Theme Development Standards

### Contents

[hide]

- [1 Why WordPress Themes](#)
- [2 Theme Development Standards](#)
  - [2.1 Anatomy of a Theme](#)
    - [2.1.1 Child Themes](#)
  - [2.2 Theme Stylesheet](#)
    - [2.2.1 Stylesheet Guidelines](#)
  - [2.3 Functions File](#)
  - [2.4 Template Files](#)
    - [2.4.1 Template Files List](#)
    - [2.4.2 Basic Templates](#)
    - [2.4.3 Custom Page Templates](#)
    - [2.4.4 Query-based Template Files](#)
    - [2.4.5 Defining Custom Templates](#)
    - [2.4.6 Including Template Files](#)
    - [2.4.7 Referencing Files From a Template](#)
    - [2.4.8 Plugin API Hooks](#)
    - [2.4.9 Untrusted Data](#)
    - [2.4.10 Translation / i18n Support](#)
    - [2.4.11 Theme Classes](#)
  - [2.5 Template File Checklist](#)
    - [2.5.1 Document Head \(header.php\)](#)
    - [2.5.2 Navigation Menus \(header.php\)](#)
    - [2.5.3 Widgets \(sidebar.php\)](#)
    - [2.5.4 Footer \(footer.php\)](#)
    - [2.5.5 Index \(index.php\)](#)
    - [2.5.6 Archive \(archive.php\)](#)
    - [2.5.7 Pages \(page.php\)](#)
    - [2.5.8 Single Post \(single.php\)](#)
    - [2.5.9 Comments \(comments.php\)](#)
    - [2.5.10 Search Results \(search.php\)](#)
  - [2.6 JavaScript](#)
  - [2.7 Screenshot](#)
  - [2.8 Theme Options](#)
- [3 Theme Testing Process](#)
- [4 Resources and References](#)
  - [4.1 Code Standards](#)
  - [4.2 Theme Design](#)
  - [4.3 CSS](#)
  - [4.4 Templates](#)
  - [4.5 Testing and QA](#)
  - [4.6 Release & Promotion](#)

[Home Page](#)

[WordPress Lessons](#)

[Getting Started](#)

[Working with WordPress](#)

[Design and Layout](#)

[Advanced Topics](#)

[Troubleshooting](#)

[Developer Docs](#)

[About WordPress](#)

### Codex Resources

[Community portal](#)

[Current events](#)

[Recent changes](#)

[Random page](#)

[Help](#)

WordPress Themes should be coded using the following standards:

- [4.7 External Resources & Tutorials](#)

- Use well-structured, error-free PHP and valid HTML. See [WordPress Coding Standards](#).
- Use clean, valid CSS. See [CSS Coding Standards](#).
- Follow design guidelines in [Site Design and Layout](#).

## Anatomy of a Theme

WordPress Themes live in subdirectories residing in `wp-content/themes/`. The Theme's subdirectory holds all of the Theme's stylesheet files, [template files](#), and optional functions file (`functions.php`), JavaScript files, and images. For example, a Theme named "test" would reside in the directory `wp-content/themes/test/`. Avoid using numbers for the theme name, as this prevents it from being displayed in the available themes list.

WordPress includes a default theme in each new installation. Examine the files in the default theme carefully to get a better idea of how to build your own Theme files.

For a visual guide, see this infographic: <http://yoast.com/wordpress-theme-anatomy/>

WordPress Themes typically consist of three main types of files, in addition to images and JavaScript files. One is the stylesheet called `style.css`, which controls the presentation (visual design and layout) of the website pages. The second is the optional functions file (`functions.php`). The other files are the [template files](#) which control the way the site pages generate the information from your WordPress database to be displayed on the site. Let's look at these individually.

## Child Themes

The simplest Theme possible is a child theme which includes only a `style.css` file, plus any images. This is possible because it is a *child* of another theme which acts as its parent.

For a detailed guide to child themes, see [Child Themes](#) or [this great tutorial by op111](#).

## Theme Stylesheet

In addition to CSS style information for your theme, `style.css` provides details about the Theme in the form of comments. The stylesheet **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 stylesheet header, for the Theme "Twenty Ten":

```
/*
Theme Name: Twenty Ten
Theme URI: http://wordpress.org/
Description: The 2010 default theme for WordPress.
Author: wordpressdotorg
Author URI: http://wordpress.org/
Version: 1.0
Tags: black, blue, white, two-columns, fixed-width, custom-header, custom-background, threaded-comments,
sticky-post, translation-ready, microformats, rtl-language-support, editor-style, custom-menu (optional)

License:
License URI:

General comments (optional).
*/
```

*NB: The name used for the Author is suggested to be the same as the Theme Author's [wordpress.org](#) username, although it can be the author's real name as well. The choice is the Theme Author's.*

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

## Stylesheet Guidelines

---

- Follow [CSS coding standards](#) when authoring your CSS.
- Use valid CSS when possible. As an exception, use vendor-specific prefixes to take advantage of CSS3 features.
- Minimize CSS hacks. The obvious exception is browsers-specific support, usually versions of IE. If possible, separate CSS hacks into separate sections or separate files.
- All possible HTML elements should be styled by the Theme, both in post/page content and in comment content.
  - Tables, captions, images, lists, block quotes, et cetera.
- Adding print-friendly styles is highly recommended.
  - You can include a print stylesheet with `media="print"` or add in a print media block in your main stylesheet.

## Functions File

---

A theme can optionally use a functions file, which resides in the theme subdirectory and is named `functions.php`. This file basically acts like a [plugin](#), and if it is present in the theme you are using, it is automatically loaded during WordPress initialization (both for admin pages and external pages). Suggested uses for this file:

- Enable [Theme Features](#) such as [Sidebars](#), [Navigation Menus](#), [Post Thumbnails](#), [Post Formats](#), [Custom Headers](#), [Custom Backgrounds](#) and others.
- Define functions used in several template files of your theme.
- Set up an options menu, giving site owners options for colors, styles, and other aspects of your theme.

The default WordPress theme contains a `functions.php` file that defines many of these features, so you might want to use it as a model. Since `functions.php` basically functions as a plugin, the [Function Reference](#) list is the best place to go for more information on what you can do with this file.

## Template Files

---

[Templates](#) are PHP source files used to generate the pages requested by visitors, and are output as HTML. 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 site. It is not essential, however, to have all these different template files for your site to fully function. 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 site. A more common use is to have different template files generate different results, to allow maximum customization.

## Template Files List

---

Here is the list of the Theme files recognized by WordPress. Of course, your Theme can contain any other stylesheets, images, or files. Just keep in mind that the following have special meaning to WordPress -- see [Template Hierarchy](#) for more information.

### **style.css**

The main stylesheet. This **must** be included with your Theme, and it must contain the information header for your Theme.

### **rtl.css**

The rtl stylesheet. This will be included **automatically** if the website's text direction is right-to-left. This can be generated using the [the RTLer](#) plugin.

### **index.php**

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

### **comments.php**

The comments template.

### **front-page.php**

The front page template, it is only used if you use a [static front page](#).

#### **home.php**

The home page template, which is the front page by default. If you use a [static front page](#) this is the template for the page with the latest posts.

#### **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.

#### **single-<post-type>.php**

The single post template used when a single post from a custom post type is queried. For example, `single-books.php` would be used for displaying single posts from the custom post type `books`. `index.php` is used if the query template for the custom post type 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.

#### **tag.php**

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

#### **taxonomy.php**

The [term template](#). Used when a term in a custom taxonomy 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 results template. Used when a search is performed.

#### **attachment.php**

Attachment template. Used when viewing a single attachment.

#### **image.php**

Image attachment template. Used when viewing a single image attachment. If not present, `attachment.php` will be used.

#### **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, according to the [Template Hierarchy](#), and when the corresponding [Conditional Tag](#) 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.

## Basic Templates

---

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

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

Both of these files go into the Theme directory. The `index.php` [template file](#) is very flexible. It can be used to include all references to the header, sidebar, footer, content, categories, archives, search, error, and any other page created in WordPress.

Or, it can be divided into modular template files, each one taking on part of the workload. If you do not provide any other template files, WordPress will use a default version. For example, if you do not have either a `comments.php` template file, WordPress will automatically use the `wp-comments.php` template files using [Template Hierarchy](#). (Note: As of version 3.0, the default files aren't guaranteed to be present or to be the same as they have been. It's much safer to supply your own template files.)

Typical template files include:

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

Using these template files you can put template tags within the `index.php` master file to include these other files where you want them to appear in the final generated page.

- To include the header, use `get_header()`.
- To include the sidebar, use `get_sidebar()`.
- To include the footer, use `get_footer()`.
- To include the search form, use `get_search_form()`.

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.

## Custom Page Templates

---

The files defining each Page Template are found in your [Themes](#) directory. To create a new Custom Page Template for a Page you must create a file. Let's call our first Page Template for our Page `snarfer.php`. At the top of the `snarfer.php` file, put the following:

```
<?php
/*
Template Name: Snarfer
*/
?>
```

The above code defines this `snarfer.php` file as the "Snarfer" Template. Naturally, "Snarfer" may be replaced with most any text to change the name of the Page Template. This Template Name will appear in the Theme Editor as the link to edit this file.

The file may be named *almost* anything with a `.php` extension (see [reserved Theme filenames](#) for filenames you should *not* use; these are special file names WordPress reserves for specific purposes).

What follows the above five lines of code is up to you. The rest of the code you write will control how Pages that use the Snarfer Page Template will display. See [Template Tags](#) for a description of the various WordPress Template functions you can use for this purpose. You may find it more convenient to copy some other Template (perhaps `page.php` or `index.php`) to `snarfer.php` and then add the above five lines of code to the beginning of the file. That way, you will only have to *alter* the HTML and PHP code, instead of creating it all from scratch. Examples are shown [below](#). Once you have created the Page Template and placed it in your Theme's directory, it will be available as a choice when you create or edit a Page. (**Note:** when creating or editing a Page, the Page Template option does not appear unless there is at least one template defined in the above manner.)

## Query-based Template Files

---

WordPress can load different [Templates](#) for different *query* types. There are two ways to do this: as part of the built-in [Template Hierarchy](#), and through the use of [Conditional Tags](#) within [The Loop](#) of a template file.

To use the [Template Hierarchy](#), you basically need to provide special-purpose Template files, which will automatically be used to override `index.php`. For instance, 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.

You can get even more specific in the Template Hierarchy by providing a file called, for instance, `category-6.php` -- this file will be used rather than `category.php` when generating the page for the category whose ID number is 6. (You can find category ID numbers in [Manage > Categories](#) if you are logged in as the site administrator in WordPress version 2.3 and below. In WordPress 2.5 the ID column was removed from the Admin panels. You can locate the category id by clicking 'Edit Category' and looking on the URL address bar for the `cat_ID` value. It will look `'...categories.php?action=edit&cat_ID=3'` where '3' is the category id). For a more detailed look at how this process works, see [Category Templates](#).

If your Theme needs to have even more control over which Template files are used than what is provided in the [Template Hierarchy](#), you can use [Conditional Tags](#). The Conditional Tag basically checks to see if some particular condition is true, within the [WordPress Loop](#), and then you can load a particular template, or put some particular text on the screen, based on that condition.

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

```
<?php
if ( is_category( '9' ) ) {
    get_template_part( 'single2' ); // looking for posts in category with ID of '9'
} else {
    get_template_part( 'single1' ); // put this on every other category post
}
?>
```

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

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

In either case, this example code will cause different templates to be used depending on the category of the particular post being displayed. Query conditions are not limited to categories, however, see the [Conditional Tags](#) article to look at all the options.

---

## 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.

---

## Including Template Files

To load another template (other than header, sidebar, footer, which have predefined included commands like `get_header()`) into a template, you can use `get_template_part()`. This makes it easy for a Theme to reuse sections of code.

---

## Referencing Files From a Template

When referencing other files within the same Theme, avoid hard-coded URIs and file paths. Instead reference the URIs and file paths with `bloginfo()`: see [Referencing Files From a Template](#).

Note that URIs that are used in the stylesheet are relative to the stylesheet, not the page that references the stylesheet. 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);
}
```

## Plugin API Hooks

---

When developing Themes, it's good to keep in mind that your Theme should be set up so that it can work well with any WordPress plugins users might decide to install. Plugins add functionality to WordPress via "Action Hooks" (see [Plugin API](#) for more information).

Most Action Hooks are within the core PHP code of WordPress, so your Theme does not have to have any special tags for them to work. But a few Action Hooks do need to be present in your Theme, in order for Plugins to display information directly in your header, footer, sidebar, or in the page body. Here is a list of the special Action Hook Template Tags you need to include:

### [wp\\_head\(\)](#)

Goes in the `<head>` element of a theme, in `header.php`. Example plugin use: add JavaScript code.

### [wp\\_footer\(\)](#)

Goes in `footer.php`, just before the closing `</body>` tag. Example plugin use: insert PHP code that needs to run after everything else, at the bottom of the footer. Very commonly used to insert web statistics code, such as Google Analytics.

### [wp\\_meta\(\)](#)

Typically goes in the `<li>Meta</li>` section of a Theme's menu or sidebar; `sidebar.php` template. Example plugin use: include a rotating advertisement or a tag cloud.

### [comment\\_form\(\)](#)

Goes in `comments.php` directly before the comment form's closing tag (`</form>`). Example plugin use: display a comment preview. As of [WordPress 3.0](#), you should use the default comment form instead, see [comment\\_form\(\)](#).

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

## Untrusted Data

---

You should escape dynamically generated content in your Theme, especially content that is output in HTML attributes. As noted in [WordPress Coding Standards](#), text that goes into attributes should be run through `esc_attr` so that single or double quotes do not end the attribute value and invalidate the XHTML and cause a security issue. Common places to check are `title`, `alt`, and `value` attributes.

In a few cases there might already be a template tag for common cases where safe output is needed. One such case involves the "title" attribute when used with `the_title()` for post and page titles. To avoid a security vulnerability, use `the_title_attribute()` instead. Here's an example of correct escaping for the `title` attribute in a post title link when using translatable text:

```
<?php echo esc_attr( sprintf( __( 'Permanent Link to %s', 'theme-name' ), the_title_attribute( 'echo=0' )
) ); ?>
```

Replace deprecated escape calls with the correct calls: `wp_specialchars`, `htmlspecialchars` with `esc_html`, `clean_url` with `esc_url`, and `attribute_escape` with `esc_attr`. See [Data Validation](#) for more.

## Translation / i18n Support

---

To ensure smooth transition for language localization, use the `gettext` functions for wrapping all translatable text within the template files. This makes it easier for the translation files to hook in and translate the titles into the site's language. See more at [WordPress Localization](#) and [i18n for WordPress Developers](#).

## Theme Classes

---

Implement the following template tags to add WordPress-generated class attributes to body, post, and comment elements. For post classes, apply only to elements within [The Loop](#).

- `body_class()`
- `post_class()`
- `comment_class()`

## Template File Checklist

---

When developing a Theme, check your template files against the following template file standards.

### Document Head (header.php)

---

- Use the proper DOCTYPE.
- The opening `<html>` tag should include `language_attributes()`.
- The "content-type" meta element should be placed before everything else, including the title element.
- Use `bloginfo()` to fetch the title and description.
- Use [automatic feed links](#) to add feed links.
- Add a call to `wp_head()`. Plugins use this action hook to add their own scripts, stylesheets, and other functionality.

Here's an example of a correctly-formatted HTML5 compliant head area:

```
<!DOCTYPE html>
<html <?php language_attributes(); ?>>
<head>
<meta charset="<?php bloginfo( 'charset' ); ?>" />
<title><?php wp_title(); ?> <?php bloginfo( 'name' ); ?></title>
<link rel="profile" href="http://gmpg.org/xfn/11" />
<link rel="stylesheet" href="<?php bloginfo( 'stylesheet_url' ); ?>" type="text/css" media="screen" />
<link rel="pingback" href="<?php bloginfo( 'pingback_url' ); ?>" />
<?php if ( is_singular() && get_option( 'thread_comments' ) ) wp_enqueue_script( 'comment-reply' ); ?>
<?php wp_head(); ?>
</head>
```

### Navigation Menus (header.php)

---

- The Theme's main navigation should support a custom menu with `wp_nav_menu()`.
  - Menu should support long link titles and a large amount of list items. These items should not break the design or layout.
  - Submenu items should display correctly. If possible, support drop-down menu styles for submenu items. Drop-downs allowing showing menu depth instead of just showing the top level.

### Widgets (sidebar.php)

---

- The Theme should be widgetized as fully as possible. Any area in the layout that works like a widget (tag cloud, blogroll, list of categories) or could accept widgets (sidebar) should allow widgets.
- Content that appears in widgetized areas by default (hard-coded into the sidebar, for example) should disappear when widgets are enabled from Appearance > Widgets.

### Footer (footer.php)

---

- Use a `wp_footer()` call, to appear just before closing body tag.

```
<?php wp_footer(); ?>
</body>
</html>
```

### Index (index.php)

---



- Display a list of posts in excerpt or full-length form. Choose one or the other as appropriate.
- Include `wp_link_pages()` to support navigation links within posts.

## Archive (archive.php)

---

- Display archive title (tag, category, date-based, or author archives).
- Display a list of posts in excerpt or full-length form. Choose one or the other as appropriate.
- Include `wp_link_pages()` to support navigation links within posts.

## Pages (page.php)

---

- Display page title and page content.
- Display comment list and comment form (unless comments are off).
- Include `wp_link_pages()` to support navigation links within a page.
- Metadata such as tags, categories, date and author should not be displayed.
- Display an "Edit" link for logged-in users with edit permissions.

## Single Post (single.php)

---

- Include `wp_link_pages()` to support navigation links within a post.
- Display post title and post content.
  - The title should be plain text instead of a link pointing to itself.
- Display the post date.
  - Respect date and time format settings unless it's important to the design. (User settings for date and time format are in Settings > General.)
  - For output based on the user setting, use `the_time( get_option( 'date_format' ) )`.
- Display the author name (if appropriate).
- Display post categories and post tags.
- Display an "Edit" link for logged-in users with edit permissions.
- Display comment list and comment form.
- Show navigation links to next and previous post using `previous_post_link()` and `next_post_link()`.

## Comments (comments.php)

---

- Author comment should be highlighted differently.
- Display avatars (user avatars) if appropriate.
- Support threaded comments.
- Display trackbacks/pingbacks.
- This file shouldn't contain function defines unless in a `function_exists()` check to avoid redeclaration errors. Ideally all functions should be in `functions.php`.

## Search Results (search.php)

---

- Display a list of posts in excerpt or full-length form. Choose one or the other as appropriate.
- The search results page show the previous search term. It's a simple but useful way to remind someone what they just searched for -- especially in the case of zero results. Use `the_search_query` or `get_search_query` (echo or return the value). For example:

```
<h2><?php printf( __( 'Search Results for: %s' ), '<span>' . get_search_query() . '</span>'); ?></h2>
```

- It's a good practice to include the search form again on the results page. Include it with: `get_search_form()`.

## JavaScript

---

- JavaScript code should be placed in external files whenever possible.

- Use `wp_enqueue_script` to load your scripts.
- JavaScript loaded directly into HTML documents (template files) should be CDATA encoded to prevent errors in older browsers.

```
<script type="text/javascript">
/*  */
// content of your Javascript goes here
/* ]]&gt; */
&lt;/script&gt;</pre></div><div data-bbox="55 170 155 186" data-label="Section-Header"><hr/><h2>Screenshot</h2></div><div data-bbox="55 195 763 239" data-label="Text"><p>Create a screenshot for your theme. The screenshot should be named <code>screenshot.png</code>, and should be placed in the top level directory. The screenshot should accurately show the theme design and saved in PNG format. The recommended image size is 300x225.</p></div><div data-bbox="55 257 190 274" data-label="Section-Header"><hr/><h2>Theme Options</h2></div><div data-bbox="55 280 697 293" data-label="Text"><p>Themes can optionally include an options page. For example code, see <a href="#">A Sample WordPress Theme Options Page</a>.</p></div><div data-bbox="55 312 780 375" data-label="Text"><p>When enabling an options page, use the <code>edit_theme_options</code> capability instead of <code>switch_themes</code> for options panels, unless the user actually should be able to switch Themes to be able to use your options panel. WordPress itself uses the <code>edit_theme_options</code> capability for menus, background, header, widgets, et cetera. See more at <a href="#">Roles and Capabilities</a> and <a href="#">Adding Administration Menus</a>.</p></div><div data-bbox="55 394 324 406" data-label="Text"><p>A note about network mode and Theme options:</p></div><div data-bbox="55 426 789 471" data-label="Text"><p>If you are using the <code>edit_themes</code> capability anywhere in your Theme, and the Theme is running on a network-enabled WordPress install (previously WordPress MU), be aware that the <code>edit_themes</code> capability used for accessing Theme options pages will prevent site admins in a network from seeing the options menu. Use <code>edit_theme_options</code> instead.</p></div><div data-bbox="55 487 300 507" data-label="Section-Header"><hr/><h2>Theme Testing Process</h2></div><div data-bbox="55 515 773 657" data-label="List-Group"><ol><li>1. Fix PHP and WordPress errors. Add the following debug setting to your <code>wp-config.php</code> file to see deprecated function calls and other WordPress-related errors: <code>define('WP_DEBUG', true);</code>. See <a href="#">Deprecated Functions Hook</a> for more information.</li><li>2. Check template files against <a href="#">Template File Checklist</a> (see above).</li><li>3. Do a run-through using the <a href="#">Theme Unit Test</a>.</li><li>4. Validate HTML and CSS. See <a href="#">Validating a Website</a>.</li><li>5. Check for JavaScript errors.</li><li>6. Test in all your target browsers. For example, IE7, IE8, IE9, Safari, Chrome, Opera, and Firefox.</li><li>7. Clean up any extraneous comments, debug settings, or TODO items.</li><li>8. See <a href="#">Theme Review</a> if you are publicly releasing the Theme by submitting it to the Themes Directory.</li></ol></div><div data-bbox="55 673 333 691" data-label="Section-Header"><hr/><h2>Resources and References</h2></div><div data-bbox="55 698 192 715" data-label="Section-Header"><hr/><h3>Code Standards</h3></div><div data-bbox="55 723 237 768" data-label="List-Group"><ul><li>▪ <a href="#">Know Your Sources</a></li><li>▪ <a href="#">WordPress Coding Standards</a></li><li>▪ <a href="#">CSS Coding Standards</a></li></ul></div><div data-bbox="55 785 183 802" data-label="Section-Header"><hr/><h3>Theme Design</h3></div><div data-bbox="55 809 204 822" data-label="List-Group"><ul><li>▪ <a href="#">Site Design and Layout</a></li></ul></div><div data-bbox="55 839 95 855" data-label="Section-Header"><hr/><h3>CSS</h3></div><div data-bbox="55 863 244 908" data-label="List-Group"><ul><li>▪ <a href="#">CSS</a></li><li>▪ <a href="#">CSS Shorthand</a></li><li>▪ <a href="#">WordPress Generated Classes</a></li></ul></div><div data-bbox="55 925 148 943" data-label="Section-Header"><hr/><h3>Templates</h3></div>
```

- [Stepping Into Templates](#)
- [Templates](#)
- [Template Hierarchy](#)
- [Template Tags](#)
- [The Loop](#)
- [Conditional Tags](#)
- [Function Reference](#)
- [I18n for WordPress Developers](#)
- [Data Validation](#)

## Testing and QA

---

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

## Release & Promotion

---

- [Theme Review Process](#)

## External Resources & Tutorials

---

- [Incorporating the Settings API in WordPress Themes](#)
- [Providing Typography Options in Your WordPress Themes](#)

Categories: [Design and Layout](#) | [WordPress Development](#) | [UI Link](#)

---

[Privacy](#) | [License / GPLv2](#) See also: [WordPress.com](#) | [WordPress.TV](#) | [WordCamp](#) | [WP Jobs](#) | [Matt](#) | [Blog RSS](#)

Like  393k

 15k