Child Themes

Languages: **English** • 日本語 • 한국어 • Français • Português do Brasil • Русский • Slovenčina • ไทย • 中文(简体) • 中文(繁體) • Македонски • (Add your language)

A WordPress child theme is a theme that inherits the functionality of another theme, called the parent theme, and allows you to modify, or add to, the functionality of that parent theme. A child theme is the safest and easiest way to modify an existing theme, whether you want to make a few tiny changes or extensive changes. Instead of modifying the theme files directly, you can create a child theme. A child theme inherits all of the templates and functionality from its parent theme, but allows you to make changes to the parent theme because code in the child theme overwrites code in the parent theme.

Why use a Child Theme?

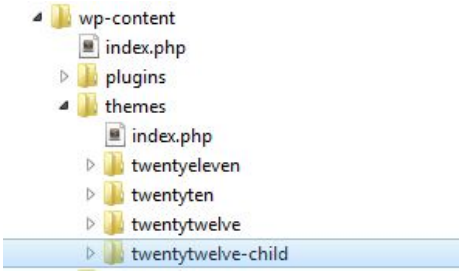
If you want to modify an existing theme, it is better to do it by creating a child theme than by modifying the parent theme directly. There are several reasons to use child themes:

- If you modify an existing theme and it is updated, your changes will be lost. With a child theme, you can update the parent theme (which might be important for security or functionality) and still keep your changes.
- It can speed up development time.
- It's a great way to get started if you are just learning WordPress theme development.

Contents

How to Create a Child Theme

- Create a directory in your themes directory to hold the child theme. The themes directory is wp-content/themes. You can name the directory whatever you want, but it is common practice to use the name of the parent theme folder with "-child" appended to it. So, for instance, if you are making a child of the twentytwelve theme, your folder name would be twentytwelve-child.
- In the child theme directory, create a file called style.css. This is the only file required to make a child theme. The style sheet must start with the following lines:




```
/*
Theme Name:     Twenty Twelve Child
Theme URI:      http://example.com/
Description:    Child theme for the Twenty Twelve theme
Author:         Your name here
Author URI:     http://example.com/about/
Template:       twentytwelve
Version:        0.1.0
*/
```

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

You can change each of these lines to suit your theme. The only required lines are the Theme Name, and the Template. The Template is the directory name of the parent theme. In this case, the parent theme is the TwentyTwelve theme, so the Template is twentytwelve, which is the name of the directory where the TwentyTwelve theme resides. If you want to make a child of a theme with the directory name some-crazy-directory-name, then you would use `Template: some-crazy-directory-name`.

- The child theme's stylesheet will overwrite the parent theme's stylesheet, but chances are you want to include the parent theme's stylesheet. To do this, you need to start the stylesheet with the following line:

```
@import url("../twentytwelve/style.css");
```

Replace `twentytwelve` with the directory name of your parent theme. This line must go after the header code and before any other CSS rules. If you put other CSS rules before the `@import`, it will not work.

- Activate the child theme. Log in to your site's dashboard, and go to Administration Panels > Appearance > Themes. You will see your child theme listed there. Click Activate.

Template Files

If you want to change more than just the stylesheet, your child theme can overwrite any file in the parent theme: simply include a file of the same name in the child theme directory, and it will overwrite the equivalent file in the parent theme directory. For instance, if you want to change the PHP code for the site header, you can include a header.php in your child theme's directory, and that file will be used instead of the parent theme's header.php.

You can also include files in the child theme that are not included in the parent theme. For instance, you might want to create a more specific template than is found in your parent theme, such as a template for a specific page or category archive. See the Template Hierarchy for more information about how WordPress decides what template to use.

Using functions.php

Unlike *style.css*, the *functions.php* of a child theme does not override its counterpart from the parent. Instead, it is **loaded in addition to the parent's functions.php**. (Specifically, it is loaded right *before* the parent's file.)

In that way, the *functions.php* of a child theme provides a smart, trouble-free method of modifying the functionality of a parent theme. Say that you want to add a PHP function to your theme. The fastest way would be to open its *functions.php* file and put the function there. But that's not smart: The next time your theme is updated, your function will disappear. But there is an alternative way which is the smart way: you can create a child theme, add a *functions.php* file in it, and add your function to that file. The function will do the exact same job from there too, with the advantage that it will not be affected by future updates of the parent theme. Do not copy the full content of functions.php of the parent theme into functions.php in the child theme.

The structure of *functions.php* is simple: An opening PHP tag at the top, a closing PHP tag at the bottom, and, between them, your bits of PHP. In it you can put as many or as few functions as you wish. The example below shows an elementary *functions.php* file that does one simple thing: Adds a favicon link to the `head` element of HTML pages.

```
function favicon_link() {
    echo '<link rel="shortcut icon" type="image/x-icon" href="/favicon.ico" />' . "\n";
}
add_action( 'wp_head', 'favicon_link' );
```

TIP FOR THEME DEVELOPERS. The fact that a child theme's *functions.php* is loaded first means that you can make the user functions of your theme pluggable —that is, replaceable by a child theme— by declaring them conditionally. E.g.:

```
if ( ! function_exists( 'theme_special_nav' ) ) {
    function theme_special_nav() {
        //  Do something.
    }
}
```

In that way, a child theme can replace a PHP function of the parent by simply declaring it beforehand.

Referencing / Including Files in Your Child Theme

When you need to include files that reside within your child theme's directory structure, you will use get_stylesheet_directory(). Because the parent template's style.css is replaced by your child theme's style.css, and your style.css resides in the root of your child theme's subdirectory, get_stylesheet_directory() points to your child theme's directory (not the parent theme's directory).

Here's an example, using `require_once`, that shows how you can use get_stylesheet_directory when referencing a file stored within your child theme's directory structure.

```
require_once( get_stylesheet_directory() . '/my_included_file.php' );
```

Other Useful Information

Using Post Formats

A child theme inherits post formats as defined by the parent theme. But, when creating child themes, be aware that using *add_theme_support('post-formats')* will **override** the formats as defined by the parent theme, not add to it.

RTL support

To support RTL languages, add **rtl.css** file to your child theme, containing:

```
/*
Theme Name: Twenty Twelve Child
Template: twentytwelve
*/

@import url("../twentytwelve/rtl.css");
```

WordPress auto-loading rtl.css file only if is_rtl(). Even if the parent theme has no rtl.css file, it's recommended to add the rtl.css file to your child theme.

Internationalization

Child themes, much like other extensions, may be prepared to be translated into other languages by using gettext functions. For an overview, please see I18n for WordPress Developers. This section will address special considerations regarding internationalization of child themes.

- Start by adding a languages directory. Something like `my-theme/languages/` works rather well.
- Next, a textdomain needs to be loaded. load_child_theme_textdomain() can be used in functions.php during the after_setup_theme action. The following should illustrate:

```
<?php
/**
 * Setup My Child Theme's textdomain.
 *
 * Declare textdomain for this child theme.
 * Translations can be filed in the /languages/ directory.
 */
function my_child_theme_setup() {
        load_child_theme_textdomain( 'my-child-theme', get_stylesheet_directory() . '/languages' );
}
add_action( 'after_setup_theme', 'my_child_theme_setup' );
?>
```

At this point, hardcoded strings in the child theme are ready to be prepared to be translated. Here is a simple example of echoing the phrase "Code is Poetry":

```
<?php
_e( 'Code is Poetry', 'my-child-theme' );
?>
```

The text domain defined in load_child_theme_textdomain() should be used to translate all strings in the child theme. In the event that a template file from the parent has been included, the textdomain should be changed from the one defined in the parent to the one defined by the child.

Resources

- Theme Development
- How to Modify WordPress Themes the Smart Way - four-part series on child themes
- How to Create a Child Theme
- Guide to WordPress Child Theme Development
- How to: Create a Child Theme Based on Twenty Eleven
- Customizing Your WordPress Theme Using Firebug
- Tutorial: Child Themes Basics and Creating Child Themes in WordPress
- How to Modify the Parent Theme Behavior Within the Child Theme
- Child Themify - a plugin for creating a child theme

Categories: Design and Layout | UI Link