# Child Themes

A child theme allows you to change small aspects of your site’s appearance yet still preserve your theme’s look and functionality. To understand how child themes work, it is first important to understand the relationship between parent and child themes.

## What is a Parent Theme?

A parent theme is a complete theme which includes all of the required WordPress template files and assets for the theme to work. All themes – excluding child themes - are considered parent themes.

## What is a Child Theme?

As indicated in the overview, a child theme inherits the look and feel of the parent theme and all of its functions, but can be used to make modifications to any part of the theme. In this way, customizations are kept separate from the parent theme’s files. Using a child theme lets you upgrade the parent theme without affecting the customizations you’ve made to your site.

**Child themes:**

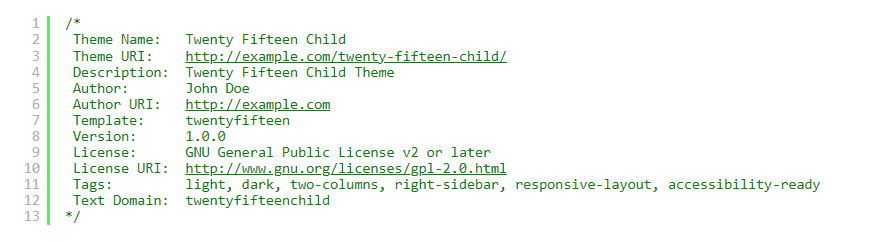
* Make your modifications portable and replicable;
* Keep customization separate from parent theme functions;
* Allow parent themes to be updated without destroying your modifications;
* Allow you to take advantage of the effort and testing put into parent theme;
* Save on development time since you are not recreating the wheel; and
* Are a great way to start learning about theme development

## How to create a Child Theme

1. **Create a child theme folder**

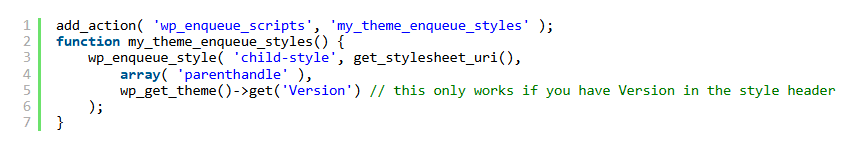
* First, create a new folder in your themes directory, located at wp-content/themes.
* The directory needs a name. It’s best practice to give a child theme at the same name as the parent, but with ***-child*** appended to the end. For example, if you were making a child theme of ***twenty fifteen***, then the directory would be named ***twenty fifteen-child.***

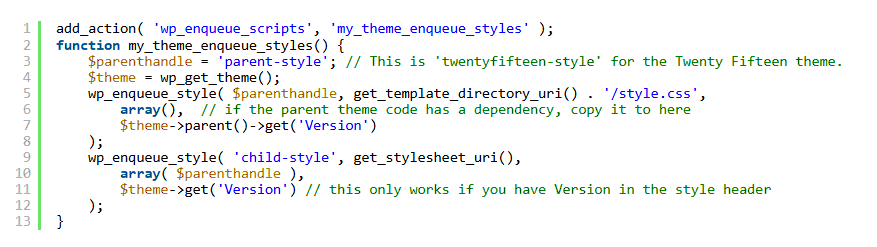
1. **Create a stylesheet: style.css**

* Next, you’ll need to create a stylesheet file named style.css, which will contain all of the CSS rules and declarations that control the look of your theme. Your stylesheet must contain the below required header comment at the very top of the file. This tells WordPress basic info about the theme, including the fact that it is a child theme with a particular parent.
* The following information is required:
* **Theme Name** – needs to be unique to your theme
* **Template** – the name of the parent theme directory. The parent theme in our example is the Twenty Fifteen theme, so the Template will be ***twenty fifteen***. You may be working with a different theme, so adjust accordingly.

*Note: Add remaining information as applicable. The only required child theme file is style.css, but functions.php is necessary to enqueue styles correctly(below).*

1. **Enqueue stylesheet**

* The final step is to enqueue the parent and child theme stylesheets, if needed.
* The ideal way of enqueuing stylesheet is for the parent theme to load both (parent’s and child’s), but not all themes do this. Therefore, you need to examine the code of the parent theme to see what it does and to get the handle name that the parent theme uses. The handle is the first parameter of **wp\_enqueue\_style().**
* There are a few things to keep in mind:
* the child theme is loaded before the parent theme.
* Everything is hooked to an action with a priority (default is 10) but the ones with the same priority run in the order they were loaded.
* For each handle, only the first call to **wp\_enqueue\_style()** is relevant (others ignored).
* The dependency parameter of **wp\_enqueue\_style()** affects the order of loading.
* Without a version number, site visitors will get whatever their browser has cached, instead of the new version.
* Using a function to get the theme’s version will return the active theme’s version (child if there is child).
* The functions named **get\_stylesheet** look for a child theme first and then the parent.
* The recommended way of enqueuing the stylesheets is to add a **wp\_enqueue\_scripts** action and use **wp\_enqueue\_style()** in your child theme’s functions.php.
* If you do not have one, create a **functions.php** in your child theme’s directory. The first line of your child theme’s **functions.php** will be a opening PHP tag (<?php), after which you can write the PHP code according to what your parent theme does.
* If the parent theme loads both stylesheets, the child theme does not need to do anything.
* If the parent theme loads its style using a function starting with **get\_template,** such as **get\_template\_directory()** and **get\_template\_directory\_uri()**, the child theme needs to load just the child styles, using the parent’s handle in the dependency parameter.
* If the parent theme loads its style using a function starting with **get\_stylesheet**, such as **get\_stylesheet\_directory()** and **get\_stylesheet\_directory\_uri()**, the child theme needs to load both parent and child stylesheets. Be sure to use the same handle name as the parent does for the parent styles.



1. **Install child theme**

* Install the child theme as you install any other theme. You can copy the folder to the site using FTP, or create a zip file of the child theme folder, choosing the option to maintain folder structure, and click on **Apppearance > Themes > Add New** to upload the zip file.

1. **Activate child theme**

* Your child theme is now ready for activation. Log in to your site’s Administration Scree, and go to **Administration Screen > Appearance > Themes.** You should see your child theme listed and ready for activation. (If your WordPress installation is multi-site enabled, then you may need to switch to your network Administration Screen to enable theme (within the Network Admin Themes Screen tab). You can then switch back to your site-specific WordPress Administration Screen to activate your child themes.)

## Adding Template Files

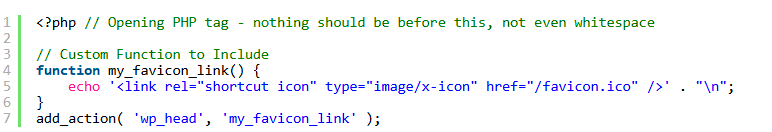
* Other than the **functions.php** file (as noted above), any file you add to your child theme will overwrite the same file in the parent theme.
* In most cases, it’s best to create a copy of the template files you want to change from the parent theme, then make your modifications to the copied files, leaving the parent files unchanged. For example, if you wanted to change the code of the parent theme’s **header.php** file, you would copy the file to your child theme folder and customize it there.
* 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 that is found in your parent theme, such as a template for a specific page or category archive (e.g. page-3.php would load for a Page with the ID of 3).

## 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 functions 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, and below it, 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.



## Referencing or Including Other Files

When you need to include files that reside within your child theme’s directory structure, you will need to use **get\_stylesheet\_directory().** Since the **style.css** is 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). To reference the parent theme directory, you would use **get\_template\_directory()** instead.

Below is the an example illustrating how to use **get\_stylesheet\_directory()** when referencing a file stored within the child theme directory:



Meanwhile, this example uses **get\_stylesheet\_directory\_uri()** to display an image that is stored within the **/images** folder in the child theme directory.



Unlike **get\_stylesheet\_directory(),** which returns a file path, **get\_stylesheet\_directory\_uri()** returns a URL, which is useful for front-end assets.

## Enqueuing Styles and Scripts

Scripts and styles should each be enqueued with their own function, and then those should be wrapped in an action.

WordPress won’t automatically load the stylesheet for your child theme on the front-end. Below is an example of using the **wp\_enqueue\_scripts()** action hook to call a function that enqueues the child theme’s stylesheet.

|  |  |
| --- | --- |
| 1  2  3  4  5 | <?php  add\_action( 'wp\_enqueue\_scripts', 'my\_plugin\_add\_stylesheet' );  **function** my\_plugin\_add\_stylesheet() {      wp\_enqueue\_style( 'my-style', get\_stylesheet\_directory\_uri() . '/style.css', false, '1.0', 'all' ); |