**The WordPress Loop Explained For Beginners**

Posted on April 22, 2014 by [Kevin Muldoon](https://www.elegantthemes.com/blog/author/kevin-muldoon) in [Tips & Tricks](https://www.elegantthemes.com/blog/category/tips-tricks) | [25 comments](https://www.elegantthemes.com/blog/tips-tricks/the-wordpress-loop-explained-for-beginners#respond)



The WordPress Loop is the code that WordPress uses to display content on your website.

Take your index.php template, for example. In a basic blogging theme, this template is generally used to display several posts or post excerpts on your home page. The index.php template uses the WordPress Loop in order to do this.

A basic understanding of the WordPress loop is necessary if you want to modify your WordPress design. Even if you do not have experience with PHP or HTML, you should be able to understand how the WordPress Loop is constructed after reading this tutorial.

**Understanding the WordPress Loop**

The best way to learn about the WordPress Loop is to look at a basic example of how it is used in a WordPress theme. Therefore, let us look at some simple code initially and then I can break down each line to give you a better understanding of what each line does.

Below is an example of a simple WordPress Loop. The code in your own WordPress theme for the loop may be much longer, however it follows the same structure as this.

|  |
| --- |
| <?php |
| if ( have\_posts() ) : | |

|  |  |
| --- | --- |
| while ( have\_posts() ) : | |
| the\_post(); |

|  |
| --- |
| // |
| // Post Content here | |

|  |
| --- |
| // |
| endwhile; // end while | |

|  |  |
| --- | --- |
| endif; // end if | |
| ?> |

If you have some experience using PHP, the above code will be self-explanatory; however, let us take a closer look at each line for the benefit of those who do not.

The first thing we do is advise your server that we are going to use [PHP](http://en.wikipedia.org/wiki/PHP). We open PHP statement by using <?php.

|  |
| --- |
| <?php |

In the next line, we have a basic “*if statement*” using the [have\_posts](https://codex.wordpress.org/Function_Reference/have_posts) function. The have\_posts WordPress function is a boolean function; which means that the result is either true of false.

Therefore, the following line of code effectively says “If there are some posts, display this line of code, if not, do nothing”.

|  |
| --- |
| if ( have\_posts() ) : |

In the next line, we use a while loop. A while loop will execute a piece of code as long as something is true. In this case, we are saying that while there are posts to be displayed, execute the following line of code.

Therefore, if you had configured your WordPress reading settings to display five posts on the home page, the while function would execute the statements contained within the while loop five times and then stop.

|  |
| --- |
| while ( have\_posts() ) : |

We then call the data from the next post by using the WordPress function [the\_post](https://codex.wordpress.org/Function_Reference/the_post). This sets up the post and allows us to retrieve any part of the post including the content, the publication date, the author, the category it was published in, and much more later on.

|  |
| --- |
| the\_post(); |

Once we have called up our post, we can display anything we want from it. There are [over one hundred template tags](https://codex.wordpress.org/Template_Tags) available that **can only be used** within the WordPress Loop.

Examples include [the\_title](https://codex.wordpress.org/Function_Reference/the_title) for displaying the post title, [the\_content](https://codex.wordpress.org/Function_Reference/the_content) for displaying the post itself, and [the\_category](https://codex.wordpress.org/Function_Reference/the_category) for displaying the post category.

|  |
| --- |
| // |
| // Post Content here | |

|  |
| --- |
| // |

After we have confirmed the information that we want displayed with every post, we close the while loop.

|  |
| --- |
| endwhile; // end while |

We then close the if statement.

|  |
| --- |
| endif; // end if |

Finally, we end by closing PHP.

|  |
| --- |
| ?> |

As you can see, when you break down the WordPress Loop, it is very easy to understand.

**Examples of the WordPress Loop**

In the WordPress codex, WordPress [shares the code](https://codex.wordpress.org/The_Loop_in_Action) for the world’s smallest index.php template. As you can see from this code (shown below), all the index.php template contains is a call to the header, the WordPress Loop, a call to a sidebar, and a call to a footer. It perfectly illustrates how simple WordPress can be.

|  |
| --- |
| <?php |
| get\_header(); | |

|  |
| --- |
| if (have\_posts()) : |
| while (have\_posts()) : | |

|  |
| --- |
| the\_post(); |
| the\_content(); | |

|  |  |
| --- | --- |
| endwhile; | |
| endif; |

|  |  |
| --- | --- |
| get\_sidebar(); | |
| get\_footer(); |

|  |
| --- |
| ?> |

The code for the WordPress Loop looks a little different in most WordPress themes because the information that is displayed is a little different. Examining the WordPress Loop in different designs is therefore a great way of seeing exactly how the loop works in practice.

Below is the code that is used for the WordPress Loop in the index.php template of the current WordPress default theme [TwentyFourteen](https://wordpress.org/themes/twentyfourteen).

As you can see, TwentyFourteen calls a different template for the content area depending on what type of post format is called e.g. a standard post will display different information from video posts and quote posts.

Navigation links are displayed after the last post and a message is displayed to visitors if no posts have been found (they do this by calling the template content-none.php).

|  |
| --- |
| <?php |
| if ( have\_posts() ) : | |

|  |
| --- |
| // Start the Loop. |
| while ( have\_posts() ) : the\_post(); | |

|  |
| --- |
|  |
| /\* | |

|  |
| --- |
| \* Include the post format-specific template for the content. If you want to |
| \* use this in a child theme, then include a file called called content-\_\_\_.php |

|  |  |
| --- | --- |
| \* (where \_\_\_ is the post format) and that will be used instead. | |
| \*/ |

|  |  |
| --- | --- |
| get\_template\_part( 'content', get\_post\_format() ); | |
|  |

|  |
| --- |
| endwhile; |
| // Previous/next post navigation. | |

|  |  |
| --- | --- |
| twentyfourteen\_paging\_nav(); | |
|  |

|  |
| --- |
| else : |
| // If no content, include the "No posts found" template. | |

|  |  |
| --- | --- |
| get\_template\_part( 'content', 'none' ); | |
|  |

|  |  |
| --- | --- |
| endif; | |
| ?> |

The WordPress Loop is easier to follow in TwentyFourteen and [TwentyThirteen](http://wordpress.org/themes/twentythirteen) because the code for content and meta information is placed in a separate template. It is also a more practical way of structuring code for themes that support post formats.

If we take a look back to the popular theme [Twenty Twelve](https://wordpress.org/themes/twentytwelve), we can see that the WordPress Loop still calls other template files; however a lot of code remains within the WordPress Loop in the index.php template.

|  |  |
| --- | --- |
| <?php if ( have\_posts() ) : ?> | |
|  |

|  |
| --- |
| <?php /\* Start the Loop \*/ ?> |
| <?php while ( have\_posts() ) : the\_post(); ?> | |

|  |  |
| --- | --- |
| <?php get\_template\_part( 'content', get\_post\_format() ); ?> | |
| <?php endwhile; ?> |

|  |
| --- |
|  |
| <?php twentytwelve\_content\_nav( 'nav-below' ); ?> | |

|  |
| --- |
|  |
| <?php else : ?> | |

|  |
| --- |
|  |
| <article id="post-0" class="post no-results not-found"> | |

|  |
| --- |
|  |
| <?php if ( current\_user\_can( 'edit\_posts' ) ) : | |

|  |  |
| --- | --- |
| // Show a different message to a logged-in user who can add posts. | |
| ?> |

|  |
| --- |
| <header class="entry-header"> |
| <h1 class="entry-title"><?php \_e( 'No posts to display', 'twentytwelve' ); ?></h1> | |

|  |  |
| --- | --- |
| </header> | |
|  |

|  |
| --- |
| <div class="entry-content"> |
| <p><?php printf( \_\_( 'Ready to publish your first post? <a href="%s">Get started here</a>.', 'twentytwelve' ), admin\_url( 'post-new.php' ) ); ?></p> | |

|  |  |
| --- | --- |
| </div><!-- .entry-content --> | |
|  |

|  |
| --- |
| <?php else : |
| // Show the default message to everyone else. | |

|  |
| --- |
| ?> |
| <header class="entry-header"> | |

|  |  |
| --- | --- |
| <h1 class="entry-title"><?php \_e( 'Nothing Found', 'twentytwelve' ); ?></h1> | |
| </header> |

|  |
| --- |
|  |
| <div class="entry-content"> | |

|  |  |
| --- | --- |
| <p><?php \_e( 'Apologies, but no results were found. Perhaps searching will help find a related post.', 'twentytwelve' ); ?></p> | |
| <?php get\_search\_form(); ?> |

|  |
| --- |
| </div><!-- .entry-content --> |
| <?php endif; // end current\_user\_can() check ?> | |

|  |
| --- |
|  |
| </article><!-- #post-0 --> | |

|  |
| --- |
|  |
| <?php endif; // end have\_posts() check ?> | |

It is best to remember the “Don’t Repeat Yourself” coding principle. By [reducing repetition](http://en.wikipedia.org/wiki/Don%27t_repeat_yourself), you can create cleaner, leaner themes that are easier to modify.

**Overview**

The WordPress Loop is used by WordPress to publish content. In its simplest form, the WordPress Loop simply checks if there are posts or pages to be displayed and then displays them.

By using multiple loops and modifying loops using the [WP\_Query](http://codex.wordpress.org/Class_Reference/WP_Query) class, theme developers can design complex website themes. The code for this is sometimes complicated, yet the general principle of how developers create complex themes is based on the basic concept of the WordPress Loop.

In this article, I have explained what the WordPress Loop is for and showed how it is used in themes to display posts and pages.

For more information on the WordPress Loop, please refer to the following pages in the WordPress codex:

* [The WordPress Loop](https://codex.wordpress.org/The_Loop)
* [The Loop in Action](https://codex.wordpress.org/The_Loop_in_Action)
* [Template Tags](https://codex.wordpress.org/Template_Tags)

I hope you have enjoyed this tutorial. Be sure to subscribe to Elegant Themes if you would like updates of our latest articles.

*Article thumbnail image by mejnak / shutterstock.com*

# [Code Reference](https://developer.wordpress.org/reference/)

Search

Browse: [Home](https://developer.wordpress.org) / [Reference](https://developer.wordpress.org/reference/) / [Functions](https://developer.wordpress.org/reference/functions/) / add\_action()

# add\_action( string $tag, callable $function\_to\_add, int $priority = 10, int $accepted\_args = 1 )

Hooks a function on to a specific action.

## Table of Contents

* [Description](https://developer.wordpress.org/reference/functions/add_action/#description)
  + [Parameters](https://developer.wordpress.org/reference/functions/add_action/#parameters)
  + [Return](https://developer.wordpress.org/reference/functions/add_action/#return)
  + [Source](https://developer.wordpress.org/reference/functions/add_action/#source)
  + [Changelog](https://developer.wordpress.org/reference/functions/add_action/#changelog)
* [More Information](https://developer.wordpress.org/reference/functions/add_action/#more-information)
  + [Usage](https://developer.wordpress.org/reference/functions/add_action/#usage)
* [Related](https://developer.wordpress.org/reference/functions/add_action/#related)
  + [Uses](https://developer.wordpress.org/reference/functions/add_action/#uses)
  + [Used By](https://developer.wordpress.org/reference/functions/add_action/#used-by)
* [User Contributed Notes](https://developer.wordpress.org/reference/functions/add_action/#user-contributed-notes)

## Description [#Description](https://developer.wordpress.org/reference/functions/add_action/#description)

Actions are the hooks that the WordPress core launches at specific points during execution, or when specific events occur. Plugins can specify that one or more of its PHP functions are executed at these points, using the Action API.

### Parameters [#Parameters](https://developer.wordpress.org/reference/functions/add_action/#parameters)

$tag

(string) (Required) The name of the action to which the $function\_to\_add is hooked.

$function\_to\_add

(callable) (Required) The name of the function you wish to be called.

$priority

(int) (Optional) Used to specify the order in which the functions associated with a particular action are executed. Lower numbers correspond with earlier execution, and functions with the same priority are executed in the order in which they were added to the action.

Default value: 10

$accepted\_args

(int) (Optional) The number of arguments the function accepts.

Default value: 1

[Top ↑](https://developer.wordpress.org/reference/functions/add_action/#top)

### Return [#Return](https://developer.wordpress.org/reference/functions/add_action/#return)

(true) Will always return true.

[Top ↑](https://developer.wordpress.org/reference/functions/add_action/#top)

### Source [#Source](https://developer.wordpress.org/reference/functions/add_action/#source)

File: [wp-includes/plugin.php](https://developer.wordpress.org/reference/files/wp-includes/plugin.php/)

|  |  |
| --- | --- |
| 457  458  459 | function add\_action($tag, $function\_to\_add, $priority = 10, $accepted\_args = 1) {      return add\_filter($tag, $function\_to\_add, $priority, $accepted\_args);  } |

[View on Trac](https://core.trac.wordpress.org/browser/tags/4.6/src/wp-includes/plugin.php#L457)