The Loop

- The Loop in Detail
  - Using The Loop
- What the Loop Can Display
- Examples
  - Basic Examples
    - Blog Archive
    - Individual Post
  - Intermediate Examples
    - Style Posts from Some Categories Differently
- Multiple Loops
  - Using rewind_posts
  - Creating secondary queries and loops
  - Resetting multiple loops
  - Using wp_reset_postdata
  - Using wp_reset_query

The Loop is the default mechanism WordPress uses for outputting posts through a theme's template files. How many posts are retrieved is determined by the number of posts to show per page defined in the Reading settings. Within the Loop, WordPress retrieves each post to be displayed on the current page and formats it according to your theme's instructions.

The Loop extracts the data for each post from the WordPress database and inserts the appropriate information in place of each template tag. Any HTML or PHP code in The Loop will be processed **for each post**.

To put it simply, the Loop is true to its name: it loops through each post retrieved for the current page one at a time and performs the action specified in your theme.

You can use the Loop for a number of different things, for example to:

- display post titles and excerpts on your blog's homepage;
- display the content and comments on a single post;

- display the content on an individual page using template tags; and
- display data from Custom Post Types and Custom Fields.

You can customize the Loop across your template files to display and manipulate different content.

The Loop in Detail

The basic loop is:

```
<?php
if ( have_posts() ) :
    while ( have_posts() ) : the_post();
        // Display post content
    endwhile;
endif;
?>
```

This loop says that when there are posts, loop through and display the posts. Broken down into more detail:

- The `have_posts()` function checks whether there are any posts.
- If there are posts, a **`while`** loop continues to execute as long as the condition in the parenthesis is logically true. As long as `have_posts()` continues to be true, the loop will continue.

Using The Loop

The Loop should be placed in `index.php`, and in any other templates which are used to display post information. Because you do not want to duplicate your header over and over, the loop should always be placed after the call to `get_header()`. For example:

```
<?php
get_header();
if ( have_posts() ) :
    while ( have_posts() ) : the_post();
        // Display post content
    endwhile;
endif;
?>
```

In the above example, the end of the Loop is shown with an `endwhile` and `endif`. The Loop must always begin with the same `if` and `while` statements, as mentioned above and must end with the same end statements.

Any template tags that you wish to apply to all posts must exist between the beginning and ending statements.

You can include a custom 404 “not found” message that will be displayed if no posts matching the specified criteria are available. The message must be placed between the `endwhile` and `endif` statements, as seen in examples below.

An extremely simple `index.php` file would look like:

```
<?php
get_header();

if ( have_posts() ) :
    while ( have_posts() ) : the_post();
        the_content();
    endwhile;
else :
    _e( 'Sorry, no posts matched your criteria.', 'textdomain' );
endif;

get_sidebar();
get_footer();
?>
```

What the Loop Can Display

The Loop can display a number of different elements for each post. For example, some common template tags used in many themes are:

- `next_post_link()` – a link to the post published chronologically *after* the current post
- `previous_post_link()` – a link to the post published chronologically *before* the current post
- `the_category()` – the category or categories associated with the post or page being viewed
- `the_author()` – the author of the post or page
- `the_content()` – the main content for a post or page
- `the_excerpt()` – the first 55 words of a post’s main content followed by an ellipsis (…) or read more link that goes to the full post. You may also use the “Excerpt” field of a post to

customize the length of a particular excerpt.
- `the_ID()` – the ID for the post or page
- `the_meta()` – the custom fields associated with the post or page
- `the_shortlink()` – a link to the page or post using the url of the site and the ID of the post or page
- `the_tags()` – the tag or tags associated with the post
- `the_title()` – the title of the post or page
- `the_time()` – the time or date for the post or page. This can be customized using standard php date function formatting.

You can also use conditional tags, such as:

- `is_home()` – Returns true if the current page is the homepage
- `is_admin()` – Returns true if inside Administration Screen, false otherwise
- `is_single()` – Returns true if the page is currently displaying a single post
- `is_page()` – Returns true if the page is currently displaying a single page
- `is_page_template()` – Can be used to determine if a page is using a specific template, for example: `is_page_template('about-page.php')`
- `is_category()` – Returns true if page or post has the specified category, for example: `is_category('news')`
- `is_tag()` – Returns true if a page or post has the specified tag
- `is_author()` – Returns true if inside author's archive page
- `is_search()` – Returns true if the current page is a search results page
- `is_404()` – Returns true if the current page does not exist
- `has_excerpt()` – Returns true if the post or page has an excerpt

Examples

Let's take a look at some examples of the Loop in action:

Basic Examples

Blog Archive

Most blogs have a blog archive page, which can show a number of things including the post title, thumbnail, and excerpt. The example below shows a simple loop that checks to see if there are any posts and, if there are, outputs each post's title, thumbnail, and excerpt. If no posts exists, it displays the message in parentheses.

```
<?php
if ( have_posts() ) :
```

```
    while ( have_posts() ) : the_post();
        the_title( '<h2>', '</h2>' );
        the_post_thumbnail();
        the_excerpt();
    endwhile;
else:
    _e( 'Sorry, no posts matched your criteria.', 'textdomain' );
endif;
?>
```

Individual Post

In WordPress, each post has its own page, which displays the relevant information for that post. Template tags allow you to customize which information you want to display.

In the example below, the loop outputs the post's title and content. You could use this example in a post or page template file to display the most basic information about the post. You could also customize this template to add more data to the post, for example the category.

```
<?php
if ( have_posts() ) :
    while ( have_posts() ) : the_post();
        the_title( '<h1>', '</h1>' );
        the_content();
    endwhile;
else:
    _e( 'Sorry, no pages matched your criteria.', 'textdomain' );
endif;
?>
```

Intermediate Examples

Style Posts from Some Categories Differently

The example below does a couple of things:

- First, it displays each post with its title, time, author, content, and category, similar to the individual post example above.
- Next, it makes it possible for posts with the category ID of "3" to be styled differently, utilizing the `in_category()` template tag.

Code comments in this example provide details throughout each stage of the loop:

```
<?php
// Start the Loop.
if ( have_posts() ) :
    while ( have_posts() ) : the_post();
        /* * See if the current post is in category 3.
          * If it is, the div is given the CSS class "post-category-three".
          * Otherwise, the div is given the CSS class "post".
        */
        if ( in_category( 3 ) ) : ?>
        <div class="post-category-three">
        <?php else : ?>
        <div class="post">
        <?php endif;

            // Display the post's title.
            the_title( '<h2>', ';</h2>' );

            // Display a link to other posts by this posts author.
            printf( __( 'Posted by %s', 'textdomain' ), get_the_author_posts_link() );

            // Display the post's content in a div.
            ?>
            <div class="entry">
                <?php the_content() ?>
             </div>

            <?php
            // Display a comma separated list of the post's categories.
            _e( 'Posted in ', 'textdomain' ); the_category( ', ' );

        // closes the first div box with the class of "post" or "post-cat-three"
       ?>
       </div>

    <?php
    // Stop the Loop, but allow for a "if not posts" situation
    endwhile;
```

```
else :
    /*
      * The very first "if" tested to see if there were any posts to
      * display. This "else" part tells what do if there weren't any.
     */
     _e( 'Sorry, no posts matched your criteria.', 'textdomain' );

// Completely stop the Loop.
 endif;
?>
```

Multiple Loops

In some situations, you may need to use more than one loop. For example you may want to display the titles of the posts in a table of content list at the top of the page and then display the content further down the page. Since the query isn't being changed we simply need to rewind the loop when we need to loop through the posts for a second time. For that we will use the function rewind_posts() .

Using rewind_posts

You can use `rewind_posts()` to loop through the *same* query a second time. This is useful if you want to display the same query twice in different locations on a page.

Here is an example of `rewind_posts()` in use:

```
<?php
// Start the main loop
if ( have_posts() ) :
    while ( have_posts() ) : the_post();
        the_title();
    endwhile;
endif;

// Use rewind_posts() to use the query a second time.
rewind_posts();
```

```
// Start a new loop
while ( have_posts() ) : the_post();
    the_content();
endwhile;
?>
```

Creating secondary queries and loops

Using two loops with the same query was relatively easy but not always what you will need. Instead, you will often want to create a secondary query to display different content on the template. For example, you might want to display two groups of posts on the same page, but do different things to each group. A common example of this, as shown below, is displaying a single post with a list of posts from the same category below the single post.

```
<?php
// The main query.
if ( have_posts() ) :
    while ( have_posts() ) : the_post();
        the_title();
        the_content();
    endwhile;
else :
    // When no posts are found, output this text.
    _e( 'Sorry, no posts matched your criteria.' );
endif;
wp_reset_postdata();

/*
 * The secondary query. Note that you can use any category name here. In our example,
 * we use "example-category".
 */
$secondary_query = new WP_Query( 'category_name=example-category' );

// The second loop.
if ( $secondary_query->have_posts() )
    echo '<ul>';
    while ( $secondary_query->have_posts() ) : $secondary_query->the_post();
        the_title( '<li>', '</li>' );
     endwhile;
     echo '</ul>';
```

```
endif;
wp_reset_postdata();
?>
```

As you can see in the example above, we first display a regular loop. Then we define a new variable that uses `WP_Query` to query a specific category; in our case, we chose the `example-category` slug.

Note that the regular loop in the example above has one difference: it calls `wp_reset_postdata()` to reset the post data. Before you can use a second loop, you need to reset the post data. There are two ways to do this:

1. By using the `rewind_posts()` function; or
2. By creating new query objects.

Resetting multiple loops

It's important when using multiple loops in a template that you reset them. Not doing so can lead to unexpected results due to how data is stored and used within the global `$post` variable. There are three main ways to reset the loop depending on the way they are called.

- `wp_reset_postdata()`
- `wp_reset_query()`
- `rewind_posts()`

Using wp_reset_postdata

Use `wp_reset_postdata()` when you are running custom or multiple loops with `WP_Query`. This function restores the global `$post` variable to the current post in the main query. If you're following best practices, this is the most common function you will use to reset loops.

To properly use this function, place the following code after any loops with `WP_Query`:

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

Here is an example of a loop using `WP_Query` that is reset with `wp_reset_postdata()`.

```
 <?php
// Example argument that defines three posts per page.
$args = array( 'posts_per_page' => 3 );

// Variable to call WP_Query.
```

```
$the_query = new WP_Query( $args );

if ( $the_query->have_posts() ) :
    // Start the Loop
    while ( $the_query->have_posts() ) : $the_query->the_post();
        the_title();
        the_excerpt();
    // End the Loop
    endwhile;
else:
// If no posts match this query, output this text.
    _e( 'Sorry, no posts matched your criteria.', 'textdomain' );
endif;

wp_reset_postdata();
?>
```

Using wp_reset_query

Using `wp_reset_query()` restores the WP_Query and global `$post` data to the original main query. You **MUST** use this function to reset your loop if you use `query_posts()` within your loop. You can use it after custom loops with WP_Query because it actually calls `wp_reset_postdata()` when it runs. However, it's best practice to use `wp_reset_postdata()` with any custom loops involving `WP_Query`.

`query_posts()` is *not best practice* and should be avoided if at all possible. Therefore, you shouldn't have much use for `wp_reset_query()`.

To properly use this function, place the following code after any loops with `query_posts()`.

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

□□□ #2
□ □□□ □□□ 2026-08-29 15:36:32 UTC
□ □□□ □□□ 2026-08-29 15:36:32 UTC