Template tag in detail:

A template tag is a PHP function used to generate and display information dynamically. WordPress Themes contain different templates and theme developers use template tags to fetch and display dynamic data. WordPress has many built-in template tags that can be used in WordPress themes. WordPress plugins and themes can also define their own template tags and use them in different templates.

Example:

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

The author template tag displays the name of the post author in WordPress.

Usage example:

```
<p>This post is written by <?php the_author(); ?></p>
```

Template tags can also return a data set and users can choose what to display using parameters.

Example:

```
<a href="<?php bloginfo('url'); ?>" title="<?php bloginfo('name'); ?>"><?php bloginfo('name'); ?></a>
```

Template tags are basically PHP functions, so any PHP function defined by a WordPress plugin or theme can be used as a template tag. To use a theme function as a template tag, the function should be defined in the theme’s functions.php file.

Template tags are PHP functions, so they can also be used inside other PHP functions and template tags. In the example below, we have defined a function that displays some text.

Example:

```
function donation_request() {

    $this_article = wp_title('',true);

    echo '<p>Hi, if you enjoyed reading '.$this_article.' please consider <a
href="http://www.example.com/donate/">donating</a>.';

}
```

To use this function in a template, add this line of code:

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

Multiple template tags can also be combined to accomplish a goal.

TEMPLATE:

In WordPress theme development, a template defines part of a web page generated by a WordPress theme.

Example: header.php is a default template used in most WordPress themes. It defines the header area of web pages generated by WordPress. The header file will typically be loaded on every page of your WordPress site, allowing changes to be made to a single file, that will apply across the entire website.

Most WordPress themes have some default templates with code to generate HTML output for particular sections of a website.

Main – index.php. To display the main page of a website.

Header – header.php. Displays the header section.

Sidebar – sidebar.php. Generates HTML output for the sidebar section.

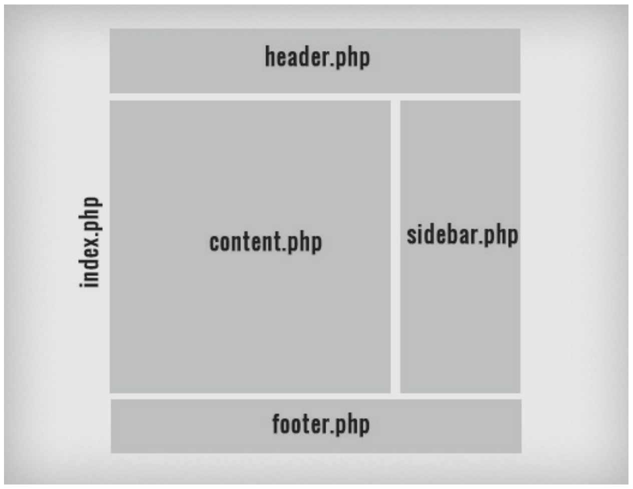
Footer – footer.php. Displays the footer section.

Theme Functions – functions.php. Contains code and functions to be used in a theme.

Single Post – single.php. Displays the single post page.

Comments – comments.php. Displays comments and comment form.

WordPress template usage example



Templates can be included into each other. For example, single.php can have header, footer, sidebar, and content template included into it. WordPress allows users to add as many as they would like. Some themes will have them for custom loops and sidebars as well.

WordPress also has a templates hierarchy. If a theme doesn't have single.php for some reason, then WordPress will automatically fall back to a more general templates like index.php.

Apart from these templates, each WordPress theme must have a style definition file named style.css.

What is theme? In detail?

A WordPress theme is a tool to change the layout and design of your website. Themes customize the appearance of your site, including the layout, typography, color, and other design elements.

A theme is a zipped folder with a group of files, including a set of page templates written in the PHP programming language combined with some CSS stylesheets.

When you first create a WordPress site, it will come with a default theme installed. You can change the theme in WordPress to match your needs and industry.

Choosing the right theme can make your website more attractive, easier to use, and improve visitor engagement.

What Is a WordPress Theme and How Does It Work?

A WordPress theme is a folder of files that work together to create the design of your site. A theme includes template files, stylesheets, images, and possibly JavaScript files. All those files change how your blog posts and pages are displayed.

Changing your WordPress theme won't necessarily change your blog content, pages, users, or other information stored in your database. Your theme only changes how all that information is displayed to your website visitors.

There are thousands of free and premium WordPress themes available for you to use.

Free themes can be downloaded from the official WordPress theme directory, or you can purchase premium themes with more advanced features from third-party developers. You can even create your own custom themes for a unique web design.

WordPress Theme vs. Template: What's the Difference?

Often the terms WordPress theme and WordPress template can confuse beginners.

Template files are part of a WordPress theme, along with other types of files like stylesheets, scripts, and images. You can't install template files individually. They work together with the rest of the files in a WordPress theme to change your website design.

A template file only controls one page or part of a page. For example, in most themes there are separate template files for your site's header, footer, sidebar, comments, and other sections. The template files help to control what information is displayed along with the layout of each page.

WordPress uses a template hierarchy that determines which template is used for which section or page.

Which Are the Most Popular WordPress Themes?

One of the first things you need to think about when you start your WordPress website is your theme.

It’s smart to select a theme that not only has the design and features you want, but is also actively updated and maintained.

The best and most popular WordPress themes available for download are:

Divi: A multi-purpose theme by Elegant Themes that’s customizable with a drag-and-drop builder

Astra: A customizable theme that comes with pre-designed demos so you can get started in minutes

Genesis by StudioPress: A clean and fast theme framework that you can style with child themes

OceanWP: A lightweight and customizable theme that’s 100% free and great for beginners

All of these themes are easy to use, well-supported, and have dozens of beautiful layouts.

They’re also all considered multi-purpose themes, which means they’ll work well for any type of website. You can use them for personal blogs, business sites, eCommerce sites, and more.

How to Install a WordPress Theme

To make your website look the way you want, you’ll need to install your chosen WordPress theme on your site.

There are three ways that you can install a WordPress theme: choose one from WordPress theme directory, upload a custom / premium WordPress, or add a new theme using FTP.

What is: Hook?

In WordPress theme and development, Hooks are functions that can be applied to an Action or a Filter in WordPress. Actions and Filters in WordPress are functions that can be modified by theme and plugin developers to change the default WordPress functionality.

Functions used to modify Actions/Filters in WordPress can be hooked into WordPress. However, it is important to note that actions and filters are not the same thing. Actions are functions performed when a certain event occurs in WordPress. Filters allow you to modify certain functions. Arguments used to hook both filters and actions look the same. But they are different in functionality and how they behave.

Example of a hook used with a filter in WordPress:

```
function wpb_custom_excerpt( $output ) {

  if ( has_excerpt() && ! is_attachment() ) {

    $output .= wpb_continue_reading_link();

  }

  return $output;

}

add_filter( 'get_the_excerpt', 'wpb_custom_excerpt' );
```

The sample code above creates a function wpb_custom_excerpt which is hooked into get_the_excerpt filter.

Example of a hook applied to an action:

```
function mytheme_enqueue_script() {

    wp_enqueue_script( 'my-custom-js', 'custom.js', false );

}

add_action( 'wp_enqueue_scripts', 'mytheme_enqueue_script' );
```

The sample code above creates a function mytheme_enqueue_script which is hooked into wp_enqueue_scripts action.

Types of hooks in detail?

What Are Action Hooks?

WordPress provides us different hooks. Using those hooks, we can efficiently execute our codes while WordPress core code executes.

It means that WordPress provides us with a Hook System, using which we can return the content that is returning as a response during the WordPress Initialization Process itself. We can change according to our needs or in the event of a specific event, i.e., in the event of a particular Action being executed, we can run a function code of our own that can meet the specific Type requirement.

The first thing is to run any of the actions. It must have the do_action() in the files. Without the do_action function, you will not add any action in the themes’ function file. do_action() has the parameter which is like below example:

```
do_action( ‘example_action_to_run’, $arg1, $arg2 );
```

Here, the example_action will be your action, and the argument you have passed you can access when you add the functions.php file.

The do_action will be majorly declared in the WordPress or WordPress plugins’ core files. As a good WordPress developer, you need to use that action rather than modifying the file directly. Let’s see how you can activate the above do_action()

```
// The action callback function.

function example_callback_fun( $arg1, $arg2 ) {

You can access the $arg1 here and $arg2 here. You just need to manipulate it and check the output.

}

add_action( ‘example_action_to_run’, ‘example_callback_fun’, 10, 2 );
```

Let me tell you all the parameters; however, it is not connected to the action and filter difference, but it is a fundamental concept which any WordPress developer needs to understand.

In the above add_action, the first parameter is the action name, the second parameter is the function name, and the third is based on the priority of the function that will run. Hence, if your priority is higher, it will run in the end. The fourth parameter is the argument. If you pass one, then you will not be able to access the second argument.

Hence, you need to check how many parameters are there in the do_action. Based on it, you can set the limit of the parameter.

There is no limitation in using the action. You can use much time, but it is based on prioritizing the last priority for the output.

What Are Filters Hooks?

Using filter hooks, we can modify content that returns while WordPress responds. Content means the text of a page or post, or author name, or any options which retrieve from the database.

Filters are functions that always accept data in the form of an Argument and, after processing again, return the data as Return Value.

The filter itself means the process of filtering something out. So, when I say that I want to filter salt from the bucket of water, the meaning is implied. So action executes something to get the results while the filter allows you to filter the data from system generated output.

Just like the do_action, here we have the apply_filters() function. If someone has not mentioned the apply_filters() in the code you must not be able to use the filter it will surely throw the error while running the page. Let’s check more in little details with examples just like the action.

```
$filtered_value = apply_filters( example_filter_to_run, ‘filter me’, $arg1, $arg2 );
```

So, now the $filtered_value will be the system generated output until you add a filter on it. In order, add the filter you need to call add_filter() just like the add_action(). Below is the example :

```
// The filter callback function based on the filter.

function example_callback_fun( $string, $arg1, $arg2 ) {

// (maybe) modify $string.

return $string;

}

add_filter( ‘example_filter_to_run’, ‘example_callback_fun’, 10, 3 );
```

Here the parameter concept remains the same, just like the add_action() function. First is the filtername. The second parameter is the function name, the third is the priority, and there the fourth is augment. Also, there is no limitation to apply filters but based on the priority; it will give the results.

Note: You need to give the return value once you modify it.

Distinguishing Differences Between Action Hooks & Filters Hooks

The primary difference between Actions Hook and Filters Hook is that Actions Hook is always rough. WordPress Action means Execute in Response to WordPress Event and does not require any type of data compulsory. Whereas Filters Hook still needs data.

Actions can have any functionality, and Filters can exist to modify data.

Actions may or may not passed any data by their action hook, and Filters are passed data to modify by their hook.

Actions do not return their changes, and Filters must return their changes.

Actions hooked in via add_action() and Filters hooked in via add_filters()

Here are some Actions Functions listed:

has_action()

do_action()

add_action()

remove_action() etc..

Here are some Filters Functions listed:

has_filter()

doing_filter()

add_filter()

remove_filter() etc..

We can categorize different Action Hooks in WordPress into different categories as follows:

Post, Page, Attachment, and Category actions

Comment. Ping and trackback actions,

Blogroll actions

Feed actions

Template actions

Administrative actions

Advanced actions

We can categorize different Filter Hooks in WordPress into different categories as follows:

Category and terms filters

Link filters

Date and Time filters

Author and User filters

Blogroll filters

Blog information and options filters

Administrative filters

Template filters

Login and Registration filters

Redirect/Rewrite filters

WP_query filters

Widget filters

Advanced WordPress filters

Conclusion

The primary use for this hook is that you do not need to change the core files or anything; you can have the default file as it so the WordPress community can handle their core files and release the update. They provide all significant hooks in the right place, which are more than enough to achieve your project requirements.In conclusion, the hooks make the WordPress developer's life easy and allow you to extend any project or task's functionality. It is allowing you to modify the content. Simultaneously, it will enable you to change the default WordPress action.

Purpose of Hooks

The primary purpose of hooks is to automatically run a function. In addition, this technique also has the ability to modify, extend, or limit the functionality of a theme or plugin.

Here is an example of a hook in WordPress:

```
function mytheme_enqueue_script()

{wp_enqueue_script( 'my-custom-js', 'custom.js', false );}

add_action( 'wp_enqueue_scripts', 'mytheme_enqueue_script' );
```

The example above shows that the hook is created to connect the mytheme_enqueue_script function with the wp_enqueue_scripts action. This hook triggers a new action on your site, therefore, it is called an action hook.

Hooks are often used in making plugin components of an application. It is not only used in content management systems (CMS) like WordPress but is also commonly used in e-commerce and intranet sites within an enterprise.

Furthermore, as we mentioned above, hooks are divided into two categories: action and filter. Action hook is used to add a process, while filter hook functions to change or modify the value of a process.

How to Use WordPress Hooks?

Using hooks in WordPress does require a bit of knowledge about HTML and PHP. However, even if you are a complete beginner, creating both action and filter hooks might not be as difficult as you think.

You only need to go to your post page, then switch to the text editor. When you are there, you can paste the hooks that you have copied from other sites or created yourself.

Creating an Action Hook

To add an action hook, you must activate the add_action () function in the WordPress plugin. This function can be activated by writing the patterns below in your functions.php file :

```
add_action('function name of target_hook', 'The_name_of_function_you_want_to_use'
 ,'priority_scale')
```

As we see above, hooks use priority scale to function properly. This scale is an automatic ordinal value based on a scale from 1 to 999. It defines the order precedence for functions associated with that particular hook.

A lower value of priority means that the function will be run earlier, while the higher one will be run later. The scale will show the output sequence of the functions installed when using the same target_hooks.

The default value of priority_scale is 10. You can arrange the scale according to the number of your target_hooks.

Here is an example of an action hook:

```
<?php

add_action( 'wp_print_footer_scripts', 'hostinger_custom_footer_scripts' );

function webhost_custom_footer_scripts(){

?>

<script>//fill the footer scripts right here</script>

<?php

}

?>
```

Note the pattern in the example above:

<?php is the place where you put the hook to function

add_action is the command for creating the action hook

wp_print_footer_scripts is the target_hook that you link to a new function

Hostinger_custom_footer_scripts is the function installed and linked to the target_hook

<script> represents the text you want to show on the target_hook (in this case, it is the wp_print_footer_scripts)

Creating a Filter Hook

You can create a filter hook by utilizing apply_filters() function. The hook filter is used to modify, filter, or replace a value with a new one.

Just like using an action hook, it also has a function that filters a value with the associated filter hook functions (apply_filter).

What is more, it has the function to add a hook filter to be associated with another function (add_filter).

Here is an example of a filter hook:

```
$score = 100;

echo "Current score is : ". apply_filters( 'change_score', $score );
```

$score = 100 is the initial score value

echo “Current score is: ” represents the script that you’re showing

apply_filters is the command to create a filter hook

‘change_score’, $score is the function to be filtered

This is the filter:

```
add_filter( 'change_score', 'function_change_score' );

function function_change_score( $score ){

? $score+=100;

? return $score;

}
```

Add_filter is created to connect the filter hook with a new function

‘Change_score‘ is the target hook that will be modified

‘Function_change_score‘ is a new function that will affect the initial value

? $score+=100; is the code to add more value to the initial value ($score)

? return $score; is the code to show the new value at the end

The result should look like this:

Current score: 200

Unhooking Actions and Filters

If you want to disable the command from add_action() or add_filter() in your WordPress code, you can use remove_action() and remove_filter().

These codes are basically a way to exclude certain actions or filter functions. It allows you to modify a plugin that has too many unnecessary hooks which might disrupt your site's optimization.

Right now, you might ask this; "why not just delete these unnecessary codes?"

Well, it certainly is a viable option if you use your own codes.

However, in WordPress, you often work with someone else's plugins or themes. It means that you risk making a fatal error if you delete the incorrect lines.

Here is an example of the remove_action()in WordPress:

```
remove_action( 'wp_print_footer_scripts', 'hostinger_custom_footer_scripts', 11 );

add_action( 'wp_print_footer_scripts', 'hostinger_custom_footer_scripts_theme', 11 );

function hostinger_custom_footer_scripts_theme()

{

?>
<script>//example of output by theme</script>

<?php

}
```

The example above shows that the remove_action is used to delete default WordPress footer scripts the replace it with Hostinger custom footer scripts theme.

This command is applicable to all kinds of action hooks in WordPress.

Furthermore, here is an example of remove_filter:

```
remove_filter( 'wp_mail', 'wp_staticize_emoji_for_email' );

add_filter( 'tiny_mce_plugins', 'disable_emojis_tinymce' );

}
```

The example above shows how to deactivate wp_staticize_emoji_for_email which converts emojis to static images.

Then it replaces them with disable_emojis_tinymce which will deactivate the emoji feature in WordPress (emoji is known to slow down the site because it makes an extra HTTP request).

Moreover, you can also use the remove_filtercommand to disable multiple filters in a sequence. Here is an example:

```
function disable_emojis() {

remove_action( 'wp_head', 'print_emoji_detection_script', 7 );

remove_action( 'admin_print_scripts', 'print_emoji_detection_script' );

remove_action( 'wp_print_styles', 'print_emoji_styles' );

remove_action( 'admin_print_styles', 'print_emoji_styles' );
remove_filter( 'the_content_feed', 'wp_staticize_emoji' );

remove_filter( 'comment_text_rss', 'wp_staticize_emoji' );

remove_filter( 'wp_mail', 'wp_staticize_emoji_for_email' );

add_filter( 'tiny_mce_plugins', 'disable_emojis_tinymce' );

add_action( 'init', 'disable_emojis' );

}
```

The code above aims to eliminate emoji function completely in WordPress. It clearly illustrates that there is no limit on how many remove_filter commands you can embed in functions.php.