What is: Template Tag

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.

The WordPress Hooks, types

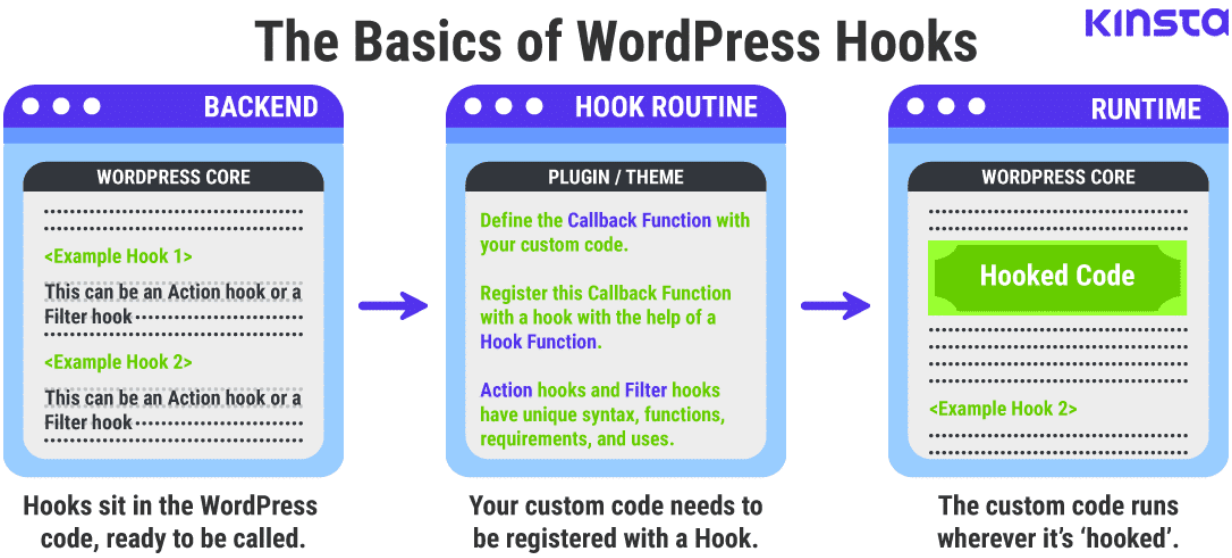
WordPress Hooks are one of the most important tools to have in a WordPress developer's arsenal. They're the foundation of WordPress plugin and theme development. You can use WordPress' many built-in hooks to 'hook into' the WordPress Core with your custom code and **do** or **modify** something.

There are two types of WordPress hooks: **Actions** and **Filters**. Hooks are so common that even WordPress Core uses them extensively itself. WordPress also includes a

way for you to define your own **custom hooks** so that other developers can hook into your code.
Learning how actions, filters, and custom hooks work is essential to master WordPress development.
The first half of this article covers the basics of WordPress hooks and explains how they work with multiple examples. In the second half, you'll learn how you can use hooks to customize WordPress, create your own custom hooks, and use them to build your own extensible plugins.

What Are WordPress Hooks?

A WordPress page is assembled by tons of functions and database queries. The WordPress Core, plugins, and theme work together to output the page elements like text, images, scripts, and styles. Once fully assembled, the browser then puts them all together and renders the page. WordPress hooks allow you to 'hook into' this build process at certain points and run your custom code. The main function of hooks is to allow you to modify or add features to WordPress without touching the core files.



Hooks will help you extend WordPress with your own code

The WordPress Plugin API powers the functionality of WordPress hooks. You use hooks by calling certain WordPress functions called **Hook Functions** at specific instances during the WordPress runtime.

Using hook functions, you can bundle your custom code within a **Callback Function** and have it registered with any hook. Once registered, this callback will run wherever the hook is, allowing you to augment or replace the default WordPress features.

The hook's position in the code execution process is an important factor. You'll learn more about its significance in the upcoming sections.

Roles and Capabilities

WordPress uses a concept of Roles, designed to give the site owner the ability to control what users can and cannot do within the site. A site owner can manage the user access to such tasks as writing and editing posts, creating Pages, creating categories, moderating comments, managing plugins, managing themes, and managing other users, by assigning a specific role to each of the users. WordPress has six pre-defined roles: Super Admin, Administrator, Editor, Author, Contributor and Subscriber. Each role is allowed to perform a set of tasks called Capabilities. There are many capabilities including "publish_posts", "moderate_comments", and "edit_users". A default set of capabilities is pre-assigned to each role, but other capabilities can be assigned or removed using the add_cap() andremove_cap() functions. New roles can be introduced or removed using the add_role() and remove_role() functions.

The Super Admin role allows a user to perform all possible capabilities. Each of the other roles has a decreasing number of allowed capabilities. For instance, the Subscriber role has just the "read" capability. One particular role should not be considered to be senior to another role. Rather, consider that roles define the user's responsibilities within the site.

Summary of Roles

- Super Admin – somebody with access to the site network administration features and all other features. See the Create a Network article.
- Administrator (*slug: 'administrator'*) – somebody who has access to all the administration features within a single site.
- Editor (*slug: 'editor'*) – somebody who can publish and manage posts including the posts of other users.
- Author (*slug: 'author'*) – somebody who can publish and manage their own posts.
- Contributor (*slug: 'contributor'*) – somebody who can write and manage their own posts but cannot publish them.
- Subscriber (*slug: 'subscriber'*) – somebody who can only manage their profile.

Upon installing WordPress, an Administrator account is automatically created.