How to Create a Site-specific WordPress Plugin

Adding custom features or functionalities to your website is often a requirement for WordPress webmasters. What you add can vary from simple lines of text to custom post types, taxonomies, shortcodes, or widgets. There are two different approaches to this matter—using third-party plugins or custom-coding the functionalities yourself. Despite the abundance of WordPress plugins, it may sometimes be necessary to custom code the features you need. Especially in cases when the plugins you use don't fully cover your requirements. And, when you've created the code you will be at a crossroads, wondering where to place that code. The two most frequently recommended places are within the functions.php file of your theme or within a site-specific plugin that you'll create. Even though the former location is more frequently recommended, we would argue that the latter is the optimal one. Let us explain why.

Firstly, any code inserted into the functions.php of your parent theme won't be usable if you deactivate the theme. Moreover, it will be lost if you update the theme, as theme files get overridden on theme updates. To mitigate this problem somewhat, you can use a child theme and insert the same code into the functions.php file that belongs to it. That way, your code will be safe during theme updates. But, the other problem remains—you will lose the additional functionality you coded if you decide to switch your theme. Therefore, using a site-specific plugin is far better, since the code is independent of the theme you are currently using, which eliminates both problems.

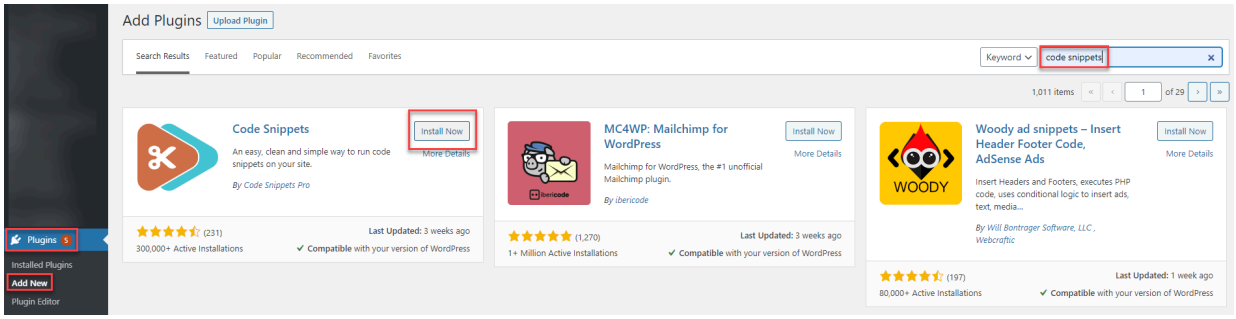
How to create a site-specific WordPress plugin

A site-specific plugin isn't something that comes by default with WordPress, it is a plugin that you need to create to insert your custom code. We will show two possible ways of achieving this—using another plugin to do code insertion or creating your plugin manually from scratch. Of the two ways mentioned, the first is much more beginner-friendly.

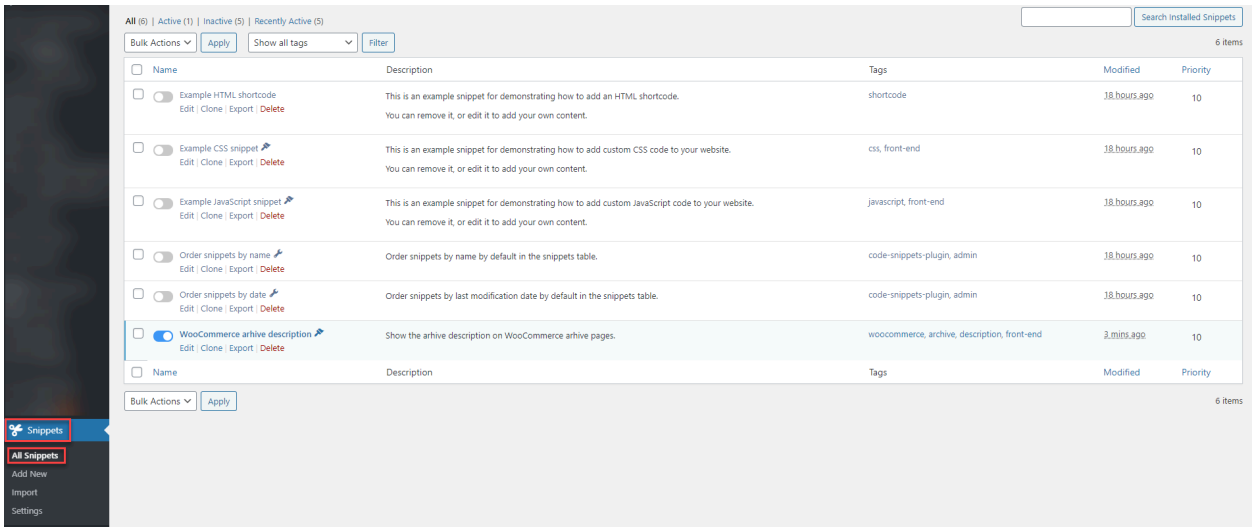
Using Code Snippets as your site-specific plugin

There are quite a few plugins you can use to insert custom code to your website, but the one we found most effective is the Code Snippets plugin.

To install it, **log in to your dashboard** and **navigate to Plugins > Add New**. **Insert *code snippets*** or a variation of that keyword into the search bar. When the appropriate plugin (you can see it on the screenshot below) appears, **press the *Install Now* button.**

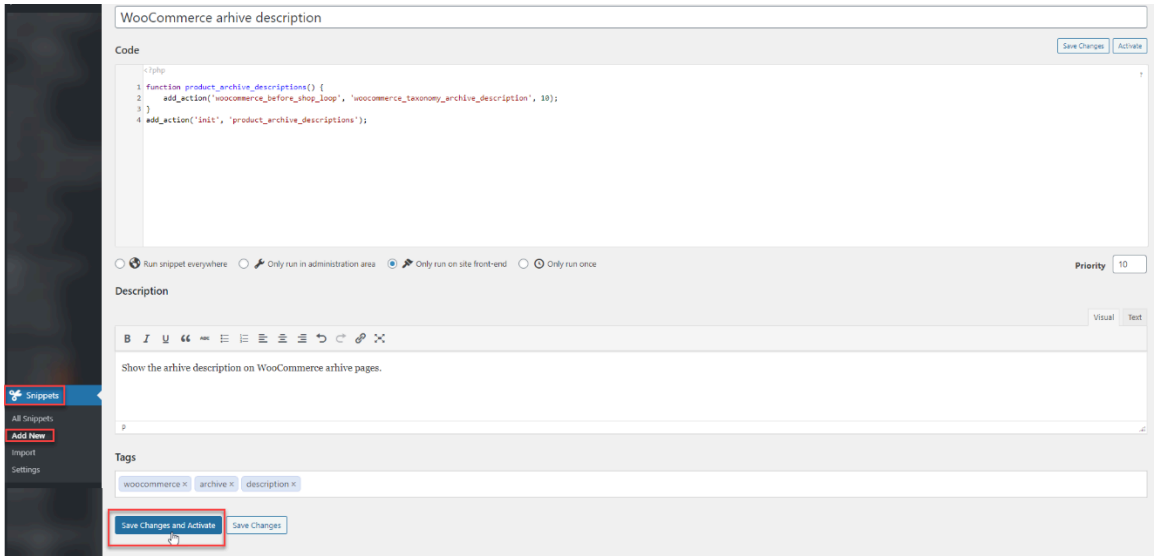


Afterward, **press the *Activate* button** and navigate to **Snippets > All Snippets**.



You will see an interface similar to the one in the Plugins section. It will have a list of snippets that you can easily activate or deactivate by clicking the activation switch next to them. Several snippets will be included with the plugin by default. These can serve as examples for your snippets. Additionally, you can order all snippets by name or date. The plugin also allows you to export your code snippets as JSON files and import them on a different website later using those same JSON files. Additionally, there is a Settings option, where you can adjust any snippet-related settings.

To add your code snippet, **navigate to Snippets > Add New**. **Insert your code** into the Code section and **choose where to run this snippet**. If you aren't sure where to run it, go with the default *Run snippet everywhere*. Then, **add a short description** of what your code snippet does in the Description section. This can prove quite useful if you have a lot of snippets inserted. Since it will take too much time to go through the code of each snippet, simply reading their descriptions can help you determine which to activate, deactivate, delete, or perform some other action on. **You should also add tags to your code snippets**. Thanks to them, you will later be able to easily find the appropriate snippet by using the tag filtering option in Snippets > All Snippets. When you've completed all the steps we suggested, **press the *Save Changes and Activate* button**.

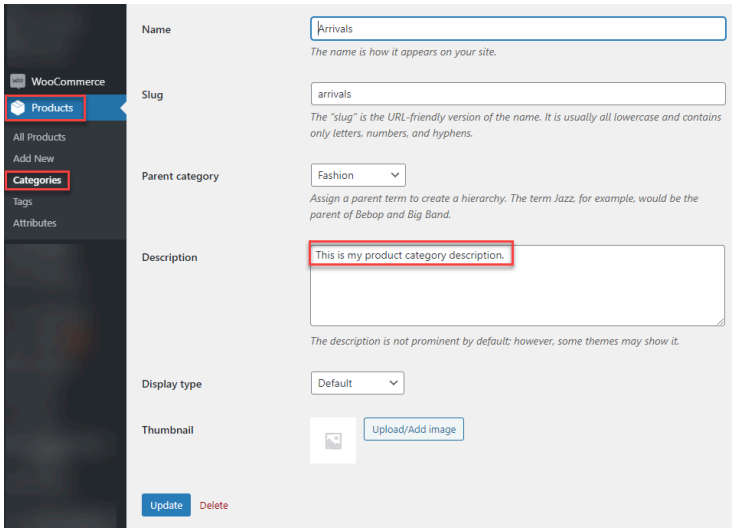


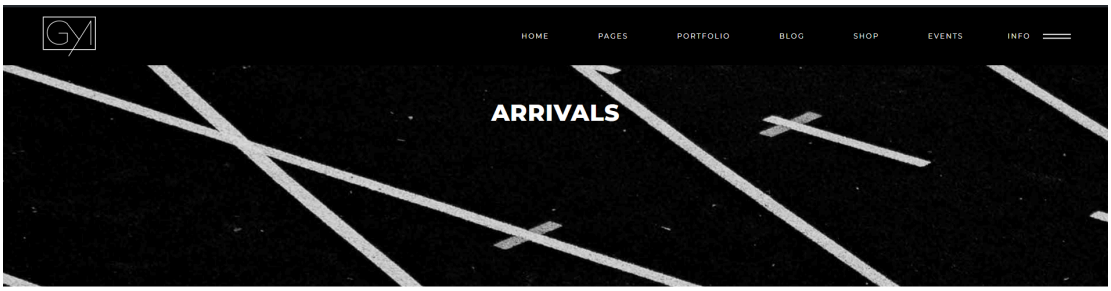
As an example, we used the following code snippet:

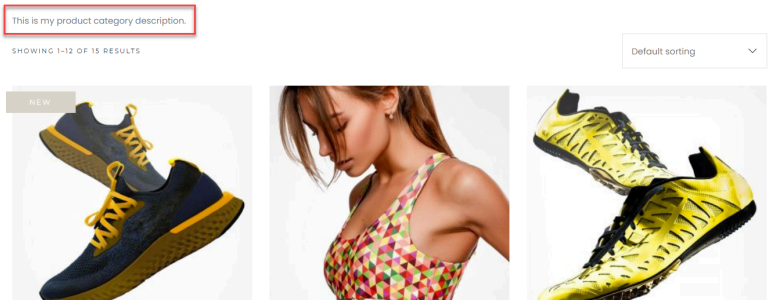
```
function product_archive_descriptions() {
        add_action('woocommerce_before_shop_loop', 'woocommerce_taxonomy_archive_description', 10);
}
add_action('init', 'product_archive_descriptions');
```

With it, you will be able to **show WooCommerce product archive description text on the top of the product archive pages (product categories and product tags)**. This small code snippet can be quite helpful for themes that don't include this feature. However, you don't need to use it for themes that already have it, as it will simply duplicate the text.

Example of how product archive descriptions would display:



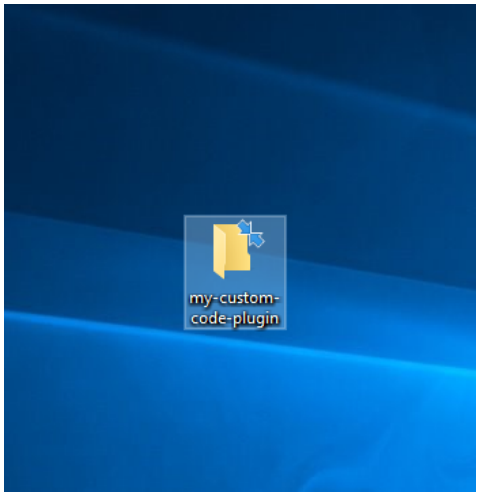




Creating your site-specific plugin manually

Since this approach is more complex, we will break down the process of creating a plugin manually into several steps. Those are creating, uploading, and editing the plugin. However, before proceeding, we highly suggest you make a backup, as creating and editing plugin files is prone to errors that can break your website.

First, **create a new folder** on the desktop of your computer. **Use lowercase and hyphens (in case of multiple words) when you name it**. For example, we have named our folder **my-custom-code-plugin**.



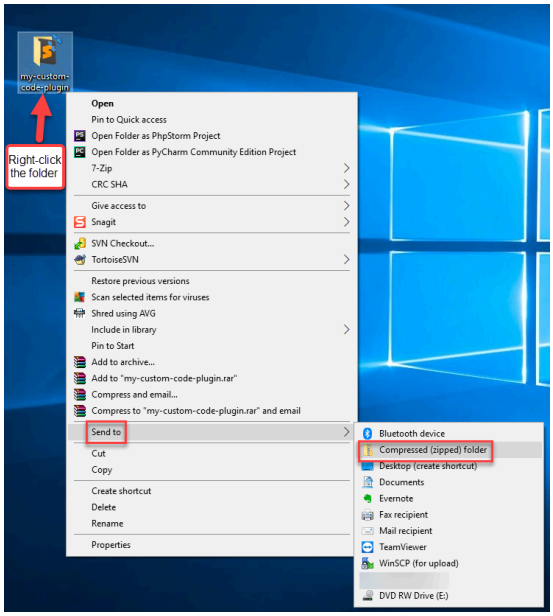
Then, **open your preferred text editor** and **create a new .php file with the same name as your folder. Then save that file within the folder you created on your desktop**. In our case, the file is called **my-custom-code-plugin.php**.

Open your new .php file using a text editor and **insert the following code** at the beginning of the file.

```
<?php

/*

Plugin Name: Custom Code Plugin for my website

Description: This plugin is made for inserting custom code specific to my website.

*/

/* Add functions below this line */


/* Stop adding functions below this line */

?>
```

These are comments, which are referred to as a plugin header. A plugin header is required for WordPress to later recognize our folder as a plugin. You can edit the comments as long as the plugin name is included. We also suggest taking a look at the plugin header link above so you can see what other header fields are available.

After inserting the plugin header into your .php file, **save the file**. Then, **locate the folder** you previously created. **Right-click on it**, **select the *Send to* option, then choose the *Compressed (zipped) folder* option**.

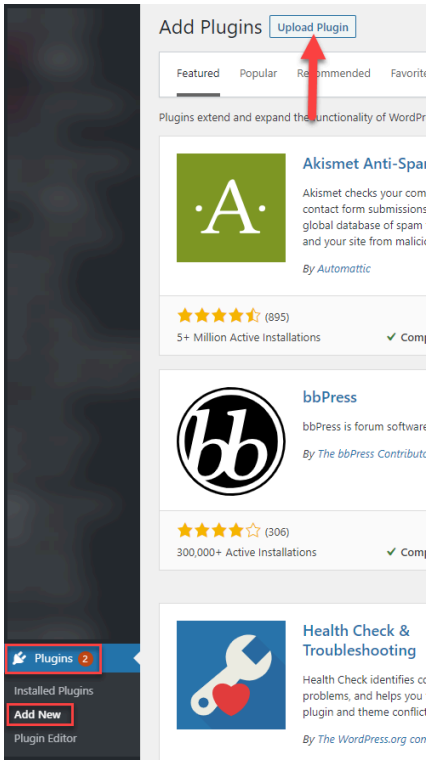


This will create a zipped file that is, in fact, your site-specific plugin.

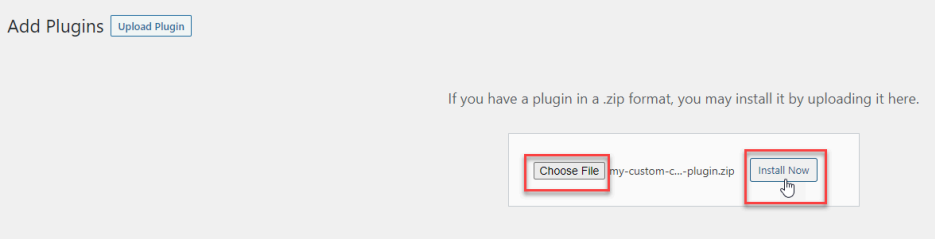
The following step is to upload the zipped file, i.e. the plugin to your website. We will show two possible ways of performing this step.

a) **Uploading your plugin via the dashboard**

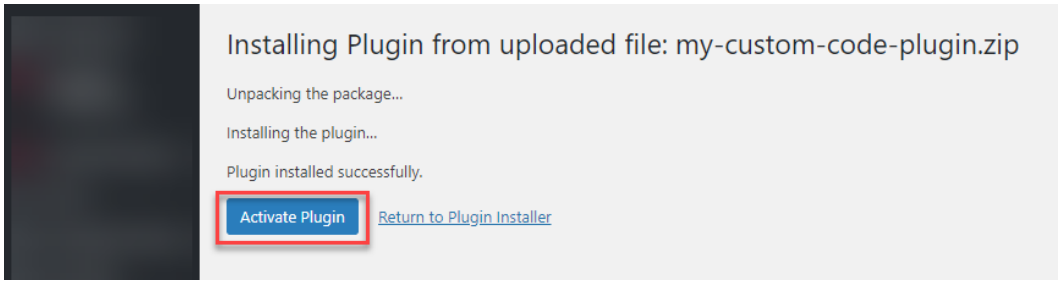
Connect to your admin dashboard using your WordPress credentials. Then, **navigate to Plugins > Add New** and **click on the *Upload Plugin* button** near the top of your screen.



Press the *Choose File* button and choose the appropriate zipped file from the upload dialog. When you're done, **press the *Install Now* button**.



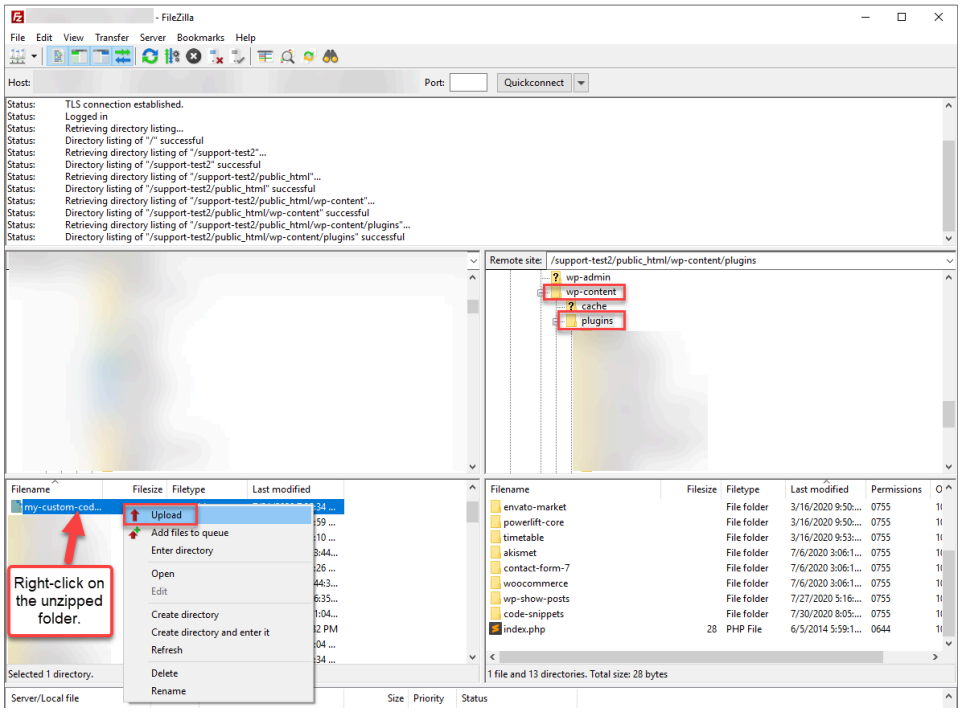
Wait until the plugin finishes installing. After the success message shows up, **press the *Activate Plugin* button**.



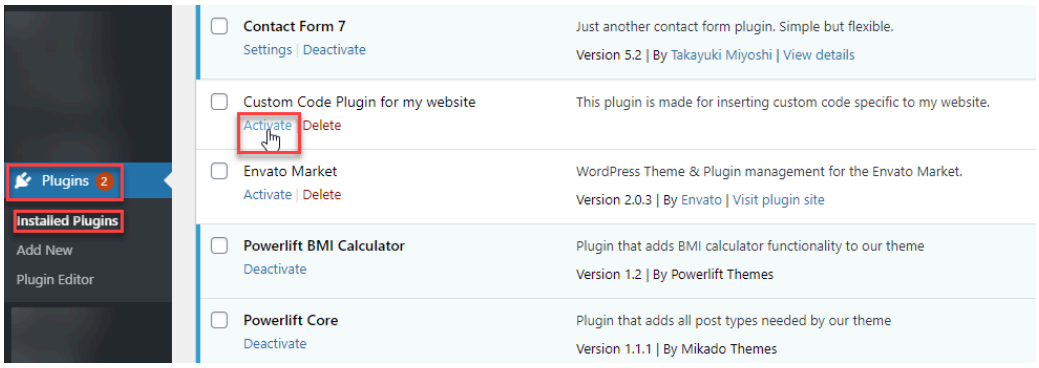
b) Uploading your plugin via FTP

To upload the plugin via FTP, **connect to your server** using your FTP credentials and **navigate to your root WordPress directory**, often called public_html. Then, **navigate to**

/wp-content/plugins/. In the left section of your FTP client **locate the unzipped plugin folder**, **right-click on it, and select the *Upload* option**.



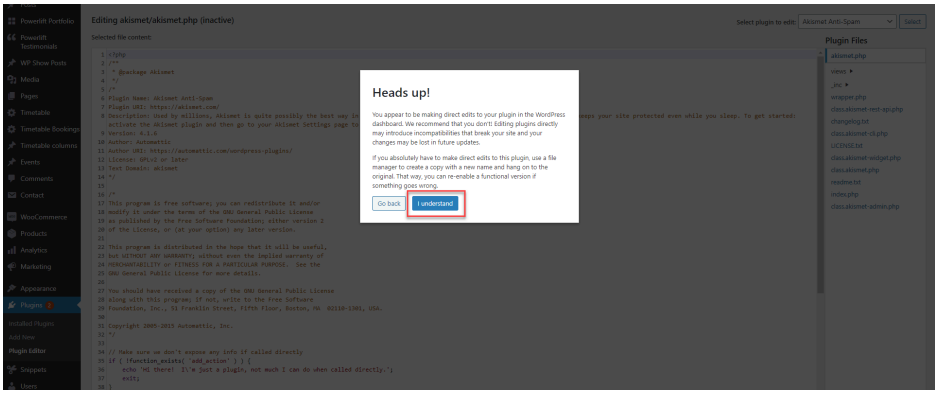
Wait a bit until the plugin folder is uploaded. Then, **log in to your dashboard** using your WordPress credentials and **navigate to Plugins > Installed Plugins**. **Locate the site-specific plugin** you just uploaded via FTP and **press the *Activate* option next to it**.



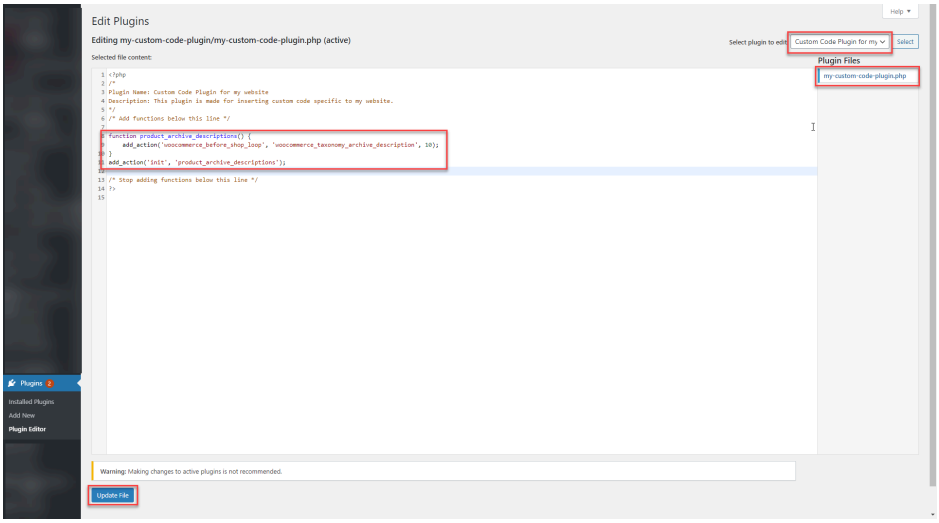
After installing the plugin, you only need to add your site-specific code. Like with the upload process, there are two ways you can do this—via the Plugin Editor within the WordPress dashboard or via FTP. We will show both.

a) Editing the plugin via the Plugin Editor

While you are logged in to your dashboard, **navigate to Plugins > Plugin Editor**. You will see a message notifying you that directly editing the plugin files could cause your site to break. If you still wish to continue, **click on the *I understand* button**.



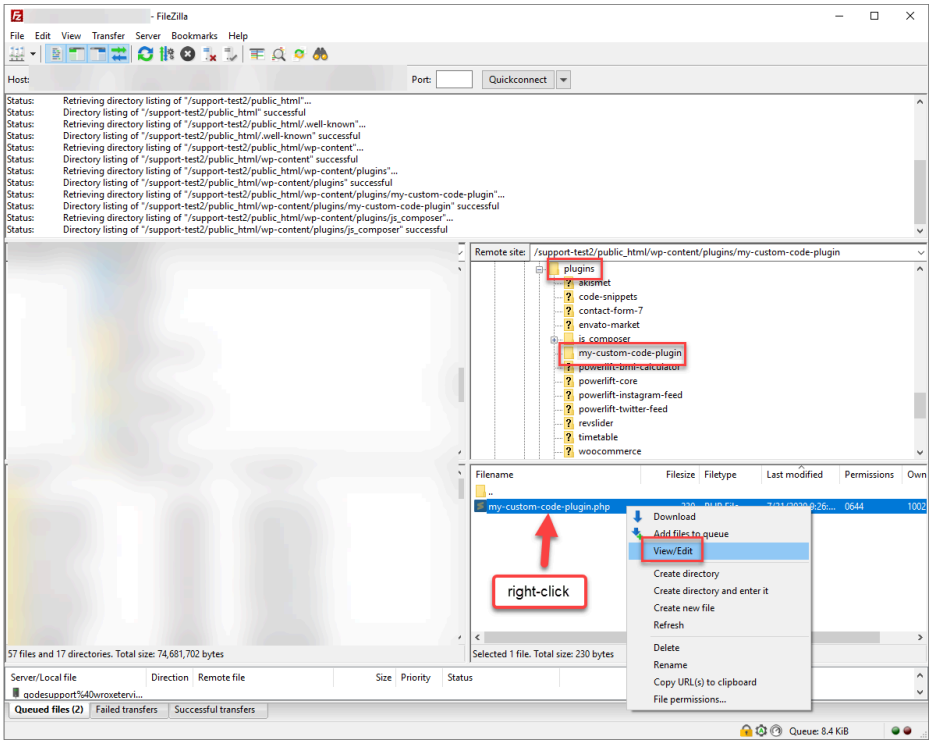
Locate your plugin among the list of plugins available for editing. **Selecting it will open the only file within that folder.** Then, **insert your code** between the two comments that mark the start and end for your code snippets, as shown in the screenshot below. **Examine your code** once more to see if there are any errors you should fix. When you're satisfied that everything is alright, **press the *Update File* button** at the bottom.



Afterward, you should get the message saying ***File edited successfully*** unless your code contained errors.

b) Editing the plugin via FTP

Connect to your server using your FTP credentials and **navigate to your root WordPress directory**, often called public_html. Then, **navigate to /wp-content/plugins/**, **find the folder of your custom plugin, and click to open it**. Within the plugin folder, **find the .php file with the same name as the plugin**. **Right-click on it, and select the *View/Edit* option** from the menu that appears.



Open the file using your preferred text editor and **insert your code between the two comment sections**, as shown in the screenshot below.

```php
<?php
/*
Plugin Name: Custom Code Plugin for my website
Description: This plugin is made for inserting custom code specific to my website.
*/
/* Add functions below this line */

function product_archive_descriptions() {
    add_action('woocommerce_before_shop_loop', 'woocommerce_taxonomy_archive_description', 10);
}
add_action('init', 'product_archive_descriptions');

/* Stop adding functions below this line */
?>
```

Afterward, **save the changes** you made and **upload the file back to your server**, thus overriding the current one.

Regarding the code, we used the same one we mentioned in the section about the Code Snippets plugin. **Its purpose is to add WooCommerce archive descriptions on product archive pages, for themes where this isn't already shown.** That code is:

```
function product_archive_descriptions() {

    add_action('woocommerce_before_shop_loop',
'woocommerce_taxonomy_archive_description', 10);
```

}

add_action('init', 'product_archive_descriptions');

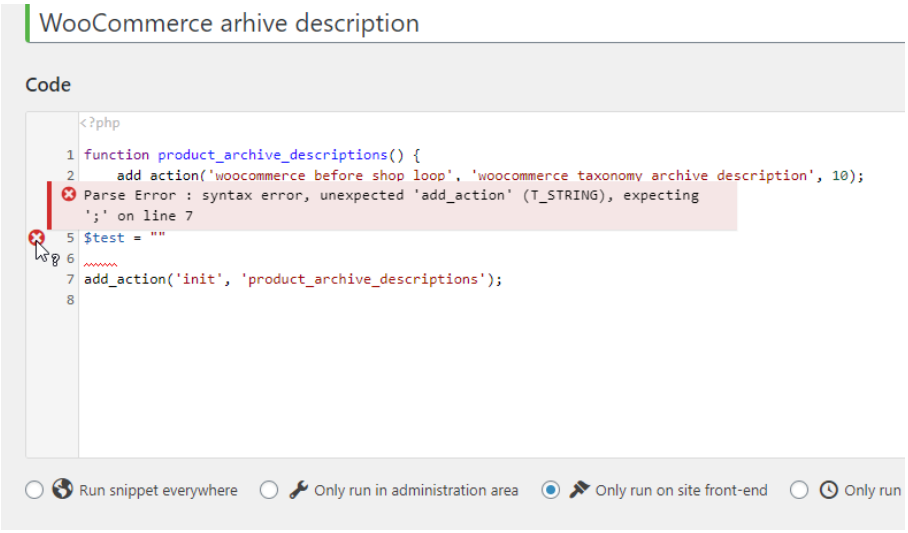
How to deal with a syntax error in your site-specific plugin

Even if you are extremely careful, code syntax errors can occur whenever you are inserting code into your website. As troubleshooting errors can be quite challenging and stressful, we included some tips on what you should be doing if the code you inserted into your site-specific plugin has a syntax error. Tips for both methods of creating site-specific plugins are included.

Using Code Snippets

Code Snippets has an in-built code checker that will notify you as soon as you make a syntax error in your code by putting a red warning sign next to the row containing the error.

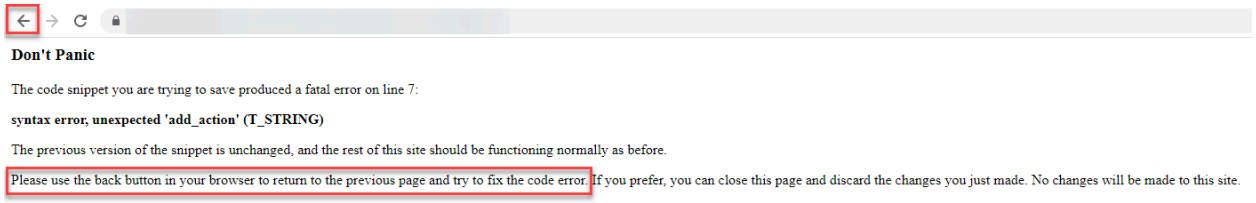
These syntax errors can be really hard to spot, as they are most often caused by a missing character in your code (opening or closing brackets, periods, or semicolons, for example). Therefore, make sure not to forget those characters. In the case that a warning sign is still present after fixing your code, **hover over it** to see the error message. This can help you determine the cause of the error.



As the error log above shows, **the error is caused by a missing semicolon (;)**, that is expected at the end of line 5. Since the semicolon isn't there to mark the end of a line, we also have a report about

unexpected code on line 7. Even though this is a simple example of a syntax error, it is helpful enough to explain how you should interpret those error messages.

However, if you were hasty and didn't check your code before pressing the *Save Changes and Activate* button, you might get an error screen that would look like the image below.



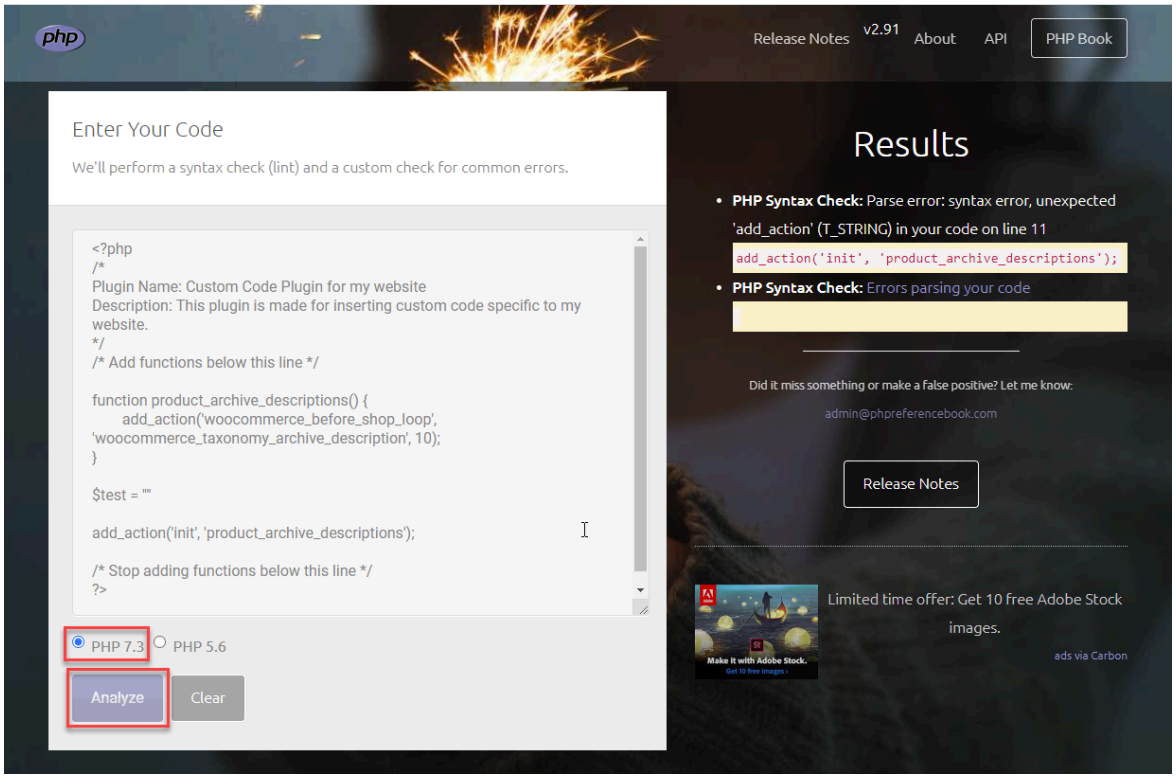
As the screen says, you shouldn't panic. Instead, simply **press the Back button in your browser** (left arrow next to the address bar in your browser, shown in the screenshot).

This will return the snippet to its previous safe copy. Meaning, if the error happened while you were editing the snippet, the snippet will be as it was before the edit. And, if the error happened because a faulty snippet was pasted without any editing, since there is no safe copy, the code will simply be removed. Then, you can try to edit it once more, while making sure there are no syntax errors this time.

Using a manually-created plugin

Inserting a code with a syntax error into your manually created plugin will cause your site to break. As such, it is important not to panic and, instead, calmly find and solve the error. To do that, you first need to **access the file with the code via FTP and open it using a text editor**. Some text editors, like PHPStorm, will have a built-in code checker (similar to the one in the Code Snippets plugin), which will make the troubleshooting much easier. **The lines containing the error will be marked red to the right of the code. Simply hover over it, and you will be able to see a helpful tip regarding the error.**

If your text editor doesn't include such code-checking features, you can always use online code-checking websites. The one we found very useful is the PHP Code Checker. Simply **copy and paste the entire code** from your .php file into their syntax-checking editor, **choose the highest available PHP version, and press the *Analyze* button**.



Afterward, **examine the results** on the right side of the screen. The error messages will point you toward the specific line of code you should look at. As we mentioned before, syntax errors are often caused by missing characters, so you should comb through your code one line at a time. However, if you need more information to solve your syntax errors, we advise learning more about common syntax mistakes.

After **you fix the errors**, **save the changes** you make within the .php file, and **upload it back to your server**, thus overwriting the faulty one.

Final thoughts

Site-specific plugins are quite helpful for storing added website functionalities. They are not dependent on the theme you are currently using, so you can add them to other sites as well. An added benefit is that tailor-made solutions can improve your website and make it more user-friendly. This is why we have done our best to carefully explain two different methods of creating a site-specific plugin, so you have options to choose from.

Furthermore, we want to say that you shouldn't be discouraged if the code you create contains errors. Trial and error, as well as problem-solving, are normal parts of working with code. To help you with that, we have included tips at the end of this article, so you have a starting point if you need to do any troubleshooting. Finally, don't forget to bookmark this article, so you can have this information at hand if you want to create additional site-specific plugins or resolve errors in the code you inserted.