# What Are WordPress Plugins?

Plugins are one of WordPress' most powerful assets. In essence, plugins are modules you activate on your website to provide a series of features or elements.

The functionality you can add to your website depends on what each specific plugin has been created to do. There are a wide selection of plugins, ranging from simple plugins (such as ones that add styling or small theme changes) all the way to extensive plugins (that provide significant changes such as eCommerce integrations or third party connections).

Plugins are different from your theme and work independently, using hooks, filters, shortcode, widgets and custom code to perform their functionality.

# Core Concepts – Actions, Filters, Shortcodes, Widgets and More

Let's take a moment to talk about some key aspects of plugin development.

You might be familiar with these areas if you have worked on WordPress themes, however, a solid understanding of how these concepts work will help you build easy to use and maintainable functionality.

# Actions (Hooks)

An `action` hook is an activity that is performed by WordPress at a specific time. When this action is triggered it will find all functions that have been attached to this action and execute them in the correct order.

WordPress has dozens of actions defined throughout its core functionality, each action consisting of a unique name. For example, when WordPress is saving a post (such as a post, page or other content type) it will call the action `save post` which will look for any functions attached to this action.

All actions are defined using the `do_action()` function. This function takes in the following parameters

- `$tag` (required)
- `$args` (optional one or more variables)

Overall, each action will have its name (its tag) and also an optional number of additional useful variables (there could be multiple or zero additional variables)

A Simple WordPress Action

Let's consider the `save_post` action hook. This action takes in four values. The first value is the name of the action, while the other three are additional variables that can be accessed (containing the id of the post, the post object itself and also if the posts exists and is being updated)

```
//the save_post action hook

do_action (`save_post`, $post_ID, $post, $update);
```

You can hook onto this action so that when a post is saved you can do something else, such as send and email or update the database.

## Using All of the Arguments (Understanding the Priority and accepted_args Values)

Sometimes using the mandatory values wont be enough. You may have to manually set the `$priority` and `$accepted_args` values to make your action work.

The `priority` of the action determines the order in which functions will be executed. By default actions will use the value of `10` for its priority. You can set this value either higher or lower to make your function execute earlier or later. This is useful when your theme or plugins are also using the same action hook (so you can specify when your function should run).

The `accepted_args` refer to how many variables are being passed in the `add_action` call. The default number of arguments that an action accepts will be one. However, if your action takes in multiple variables you have to tell WordPress how many it is taking. Let's look at the `save_post` action.

```
//add our function onto the 'save_post' hook, supplying priority and args

add_action('save_post','save_my_page_extended,10,3);

//function executed on save (we want all three variables)

function save_my_page_extended($post_ID, $post, $update){

    //access all of our variables to do some work

}
```

The `save_post` action has multiple variables that can be passed to it, as such we have to set its priority and also the number of variables it will be passing. Once we have told WordPress we will be accepting multiple variables, we can add these into our function and we will be able to access them.

## Filters (Hooks)

A WordPress `filter` is a hook that accepts a variable (or series of variables) and returns them back after they have been modified. These filters are often used so that you have a way to manipulate default information.

WordPress comes bundled with dozens of these filters and they are created using the `apply_filters()` function. This function takes in the following arguments

## A Simple WordPress Filter

The `get_the_excerpt` filter is a filter that you can use inside the posts loop to access the excerpt.

This filter is defined as part of WordPress's core and only defines the name of the filter and the function to call, it doesn't define any optional arguments.

```
//get the except for a post, as defined in /wp-includes/post-template.php

echo apply_filters( 'the_excerpt', get_the_excerpt() );
```

You could attach your own function to the `the_excerpt` filter and then manipulate the excerpt before you return it (for example, you could wrap it inside a container, change its content or trim its length).

## Shortcodes

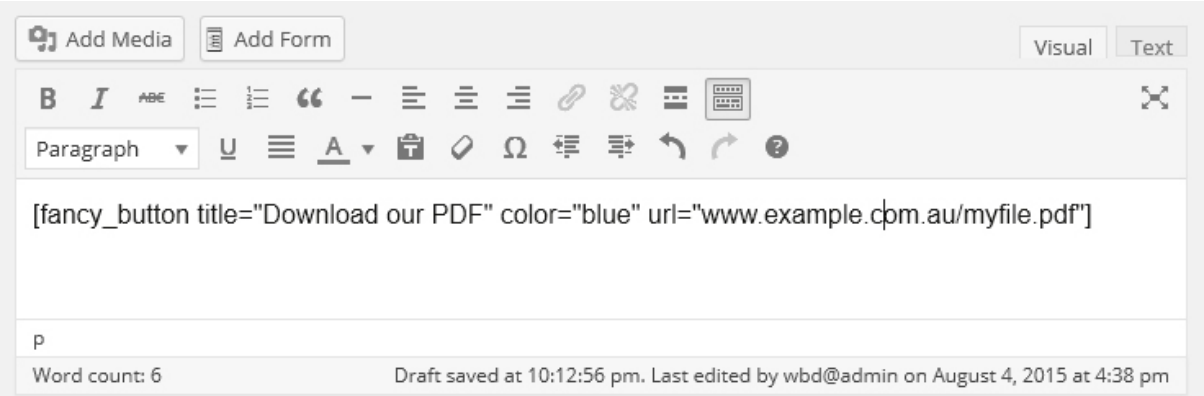
Since your plugin generally won't have access to your theme files, you need a way to display your custom functionality to the end user while making it easy to customize by your site admin.

Shortcodes are perfect way to do this as they allow developers to create complex functionality that can be modified with simple values and attributes. These shortcodes are added to the visual editor (or other locations) and when viewed by your users they execute your code.

Let's look at a simple example:

```php
//Add our shortcode

add_shortcode('test_shortcode','my_shortcode_output');

//perform the shortcode output

function my_shortcode_output($atts, $content = '', $tag){

    $html = '';

    $html .= '<p>Hello World</p>';

    return $html;

}
```

Now when we add the shortcode `[test_shortcode]` to the editor it will run our function and convert the results to `<p>Hello World</p>`




For real development of plugins watch this tutorial:

https://www.youtube.com/watch?v=Z7QfH-s-15s