add_action(string $hook_name, callable $callback, int $priority = 10, int $accepted_args = 1): true

Adds a callback function to an action hook.

Description

Actions are the hooks that the WordPress core launches at specific points during execution, or when specific events occur. Plugins can specify that one or more of its PHP functions are executed at these points, using the Action API.

Parameters

`$hook_name`stringrequired
The name of the action to add the callback to.

`$callback`callablerequired
The callback to be run when the action is called.

`$priority`intoptional
Used to specify the order in which the functions associated with a particular action are executed.
Lower numbers correspond with earlier execution, and functions with the same priority are executed in the order in which they were added to the action.

Default: `10`

`$accepted_args`intoptional
The number of arguments the function accepts.

Default: `1`

Return

`true` Always returns true.

More Information

Usage

wp-includes/plugin.php

Copy

```
add_action( $hook, $function_to_add, $priority, $accepted_args );
```

To find out the number and name of arguments for an action, simply search the code base for the matching do_action() call. For example, if you are hooking into 'save_post', you would find it in post.php:

wp-includes/plugin.php

Copy

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

Your add_action call would look like:

wp-includes/plugin.php

Copy

```
add_action( 'save_post', 'wpdocs_my_save_post', 10, 3 );
```

And your function would be:

wp-includes/plugin.php

Copy

```
function wpdocs_my_save_post( $post_ID, $post, $update ) {
// do stuff here
}
```

Source

wp-includes/plugin.php

Copy

```
function add_action( $hook_name, $callback, $priority = 10, $accepted_args = 1 ) {
    return add_filter( $hook_name, $callback, $priority, $accepted_args );
}
```

View all references·View on Trac·View on GitHub

Related

Uses

add_filter()
wp-includes/plugin.php

Used by

WP_Interactivity_API::data_wp_router_region_processor()
wp-includes/interactivity-api/class-wp-interactivity-api.php

WP_Interactivity_API::add_hooks()
wp-includes/interactivity-api/class-wp-interactivity-api.php

WP_Script_Modules::add_hooks()
wp-includes/class-wp-script-modules.php

WP_Textdomain_Registry::init()
wp-includes/class-wp-textdomain-registry.php

wp_initialize_theme_preview_hooks()
wp-includes/theme-previews.php

Show 83 more

Changelog

Version	Description
1.2.0	Introduced.

User Contributed Notes

Codex9 years ago 46

Using with a Class

To use `add_action()` when your plugin or theme is built using classes, you need to use the array callable syntax. You would pass the function to `add_action()` as an array, with `$this` as the first element, then the name of the class method, like so:

Expand code Copy

```
/**
 * Class WP_Docs_Class.
 */
class WP_Docs_Class {

    /**
     * Constructor
     */
    public function __construct() {
        add_action( 'save_post', array( $this, 'wpdocs_save_posts' ) );
    }

    /**
```

Show feedback (1)Log in to add feedback

Codex9 years ago 25

Using with static functions in a class

If the class is called staticly the approach has to be like below as `$this` is not available. This also works if class is extended. Use the following:

Expand code Copy

```
/**
 * Class WP_Docs_Static_Class.
 */
class WP_Docs_Static_Class {
    /**
     * Initializer for setting up action handler
     */
    public static function init() {
        add_action( 'save_post', array( get_called_class(), 'wpdocs_save_posts' ) );
    }

    /**
```

Show feedback (1)Log in to add feedback

Codex9 years ago 10

Simple Hook

To email some friends whenever an entry is posted on your blog:

Copy

```
/**
 * Send email to my friends.
 *
 * @param int $post_id Post ID.
 * @return int Post ID.
 */
function wpdocs_email_friends( $post_id ) {
    $friends = 'bob@example.org, susie@example.org';
    wp_mail( $friends, "sally's blog updated", 'I just put something on my blog: http://blog.example

    return $post_id;
}
add_action( 'publish_post', 'wpdocs_email_friends' );
```

Log in to add feedback

mkormendy4 years ago 9

To pass a variable to the called function of the action, you can use closures (since PHP 5.3+) when the argument is not available in the original coded do_action. For example:

Copy

```
add_action('wp_footer', function($arguments) use ($myvar) {
    echo $myvar;
}, $priority_integer, $accepted_arguments_integer);
```

Show feedback (1)Log in to add feedback

Anthony Hortin7 years ago 5

Related:

do_action()
remove_action()

Log in to add feedback

lucasbustamante6 years ago 3

Passing parameters while using in a Class
To pass parameters to your method in a Class while calling it with add_action, you can do as following:

Expand code Copy

```
public function __construct() {
    // Actions
    add_action('init', array($this, 'call_somefunction'));
}

/**
 *    Intermediate function to call add_action with parameters
 */
public function call_somefunction() {
    $this->somefunction('Hello World');
}

/**
```

Log in to add feedback

Christian Saborio4 years ago 2

How to add an action that calls a function (with parameters) from an instantiated class:

Copy

```
$admin_menu_hider = new AdminMenuHider( UserManagement::get_internal_users() );
            add_action(
                'wp_before_admin_bar_render',
                function () use ( $admin_menu_hider ) {
                    $admin_menu_hider->change_greeting_message( 'Hello' );
                }
            );
```

Log in to add feedback

Bartek1 year ago 2

I urge you, don’t attach your hook callbacks inside class’ constructor.

Instead of implementing official example most upvoted in this thread, opt for decoupled solution. You have one more line of code to write, but objects become more reusable and less error-prone (consider, what would happen if you call `new WP_Docs_Class()` twice in your code, following Codex example).

Expand code Copy

```
/**
 * Class WP_Docs_Class.
 */
class WP_Docs_Class {

    /**
     * Initiate all hooks' callbacks in a separate method.
     */
    public function hooks() {
        add_action( 'save_post', array( $this, 'wpdocs_save_posts' ) );
    }

    /**
```

Log in to add feedback

Codex9 years ago 1

Accepted Arguments
A hooked function can optionally accept arguments from the action call, if any are set to be passed. In this simplistic example, the `echo_comment_id` function takes the `$comment_id` argument, which is automatically passed to when the `do_action()` call using the `comment_id_not_found` filter hook is run.

Copy

```
/**
 * Warn about comment not found
 *
 * @param int $comment_id Comment ID.
 */
function echo_comment_id( $comment_id ) {
    printf( 'Comment ID %s could not be found', esc_html( $comment_id ) );
}
add_action( 'comment_id_not_found', 'echo_comment_id', 10, 1 );
```

Log in to add feedback

theking21 year ago -4

To prevent runtime errors due to easy typo errors (defensive programming) and prevent pollution of the global namespace use closures instead of function names. The sample code adjusted:

Copy

```
/**
 * add a save post hook
 */
add_action( 'save_post', function( $post_ID, $post, $update ) {
   // do stuff here
}, 10, 3 );
```

Show feedback (2)Log in to add feedback

You must log in before being able to contribute a note or feedback.