Advanced Topics

- Removing Actions and Filters
- Example
- Removing All Callbacks
- Determining the Current Hook
- Checking How Many Times a Hook Has Run
- Debugging with the "all" Hook

Removing Actions and Filters

Sometimes you want to remove a callback function from a hook that another plugin, theme or even WordPress Core has registered.

To remove a callback function from a hook, you need to call `remove_action()` or `remove_filter()`, depending whether the callback function was added as an Action or a Filter.

The parameters passed to `remove_action()` / `remove_filter()` must be identical to the parameters passed to `add_action()` / `add_filter()` that registered it, or the removal won't work.

[alert]To successfully remove a callback function you must perform the removal after the callback function was registered. The order of execution is important.[/alert]

Example

Let's say we want to improve the performance of a large theme by removing unnecessary functionality.

Let's analyze the theme's code by looking into `functions.php`.

```
function wporg_setup_slider() {
  // ...
```

```
}
add_action( 'template_redirect', 'wporg_setup_slider', 9 );
```

The `wporg_setup_slider` function is adding a slider that we don't need, which probably loads a huge CSS file followed by a JavaScript initialization file which uses a custom written library the size of 1MB. We can can get rid of that.

Since we want to hook into WordPress after the `wporg_setup_slider` callback function was registered (`functions.php` executed) our best chance would be the `after_setup_theme` hook.

```
function wporg_disable_slider() {
  // Make sure all parameters match the add_action() call exactly.
  remove_action( 'template_redirect', 'wporg_setup_slider', 9 );
}
// Make sure we call remove_action() after add_action() has been called.
add_action( 'after_setup_theme', 'wporg_disable_slider' );
```

Removing All Callbacks

You can also remove all of the callback functions associated with a hook by using `remove_all_actions()` / `remove_all_filters()`.

Determining the Current Hook

Sometimes you want to run an Action or a Filter on multiple hooks, but behave differently based on which one is currently calling it.

You can use the `current_action()` / `current_filter()` to determine the current Action / Filter.

```
function wporg_modify_content( $content ) {
  switch ( current_filter() ) {
    case 'the_content':
      // Do something.
      break;
    case 'the_excerpt':
      // Do something.
      break;
  }
```

```
  return $content;
}

add_filter( 'the_content', 'wporg_modify_content' );
add_filter( 'the_excerpt', 'wporg_modify_content' );
```

Checking How Many Times a Hook Has Run

Some hooks are called multiple times in the course of execution, but you may only want your callback function to run once.

In this situation, you can check how many times the hook has run with the `did_action()`.

```
function wporg_custom() {
  // If save_post has been run more than once, skip the rest of the code.
  if ( did_action( 'save_post' ) !== 1 ) {
    return;
  }
  // ...
}
add_action( 'save_post', 'wporg_custom' );
```

Debugging with the "all" Hook

If you want a callback function to fire on every single hook, you can register it to the `all` hook. Sometimes this is useful in debugging situations to help determine when a particular event is happening or when a page is crashing.

```
function wporg_debug() {
  echo '<p>' . current_action() . '</p>';
}
add_action( 'all', 'wporg_debug' );
```

□□□ #2
□ □□□ □□□ 2026-10-26 15:37:07 UTC
□ □□□ □□□ 2026-10-26 15:37:07 UTC