Nathan Rice

WordPress Themes, Plugins, and Web Development

An Introduction to WordPress Action Hooks

June 10, 2009 by Nathan Rice (http://www.nathanrice.net/blog/author/nathanrice/)

If you're going to be doing *any* level of WordPress development, *themes or plugins*, you will invariably run into the need to take advantage of the **WordPress Action Hook** system. **But the more I am able to talk to people in the community, the more I realize that people simply don't understand the concept very well, or at all.**

In fact, one of the biggest barriers to using what is referred to as a Theme Framework (http://codex.wordpress.org/Theme_Frameworks) or creating their own plugins is the fact that they rely heavily on Action Hooks to function properly.

Understanding this concept accelerated my level of development skills immediately after I figured out what hooks were, and how they worked. And today, I want to help you find that path too.

So, What Are WordPress Action Hooks?

This can be a very easy, and a somewhat difficult thing to explain. **Action hooks are essentially placeholders. Wherever an action hook is placed, it will execute any code that has been "hooked" to it.**

So, let's try to visualize this with some default WordPress action hooks that are in most themes. You can find `wp_head` and `wp_footer` in just about every single theme available, and most people don't realize **these are action hooks**. They're simply placeholders that

plugins can use to insert code into the `<head>` and footer of the theme. Often times, they use these action hooks to insert things like `CSS` or Analytics code. They create a function that generates the code, and then "hook" that function to either `wp_head` or `wp_footer`.

If I could wrap up the concept in one sentence, it would be this: **WordPress action hooks are a means of providing a way for other developers to insert their own code in specific locations within your code, in order to change or expand the functionality of your code.**

Why Action Hooks Are Necessary

Like any other major piece of software, WordPress evolves. Every few months a new version is released.

Let's assume that action hooks didn't exist, but you wanted to change or extend some function of WordPress. In order to do this, you have to modify core WordPress files. And when it comes time to upgrade, you're left with a choice: do the upgrade and lose all my modifications, or stay with the old version.

But **if you use action hooks to modify how WordPress works, you can upgrade knowing that your mods are in a separate file which makes the changes the correct way, and you won't be in danger of having those mods overwritten during the upgrade.**

How Do Action Hooks Work?

Going back to the example of the `wp_head` and `wp_footer` actions, let's assume that you wanted to insert some `CSS` in your `<head>` section of your theme so that you can override the link colors.

(don't worry about the "what" so much. the "how" is much more important here).

The first thing you need to do is create a function that inserts the code. Because action hooks only allow functions to be "hooked" to them, we'll need to create a function that generates the code:

```
function insert_some_css() {
        echo <<<CSS
<style type="text/css">
```

```
        a {
                color: #08FF00; /* green */
                text-decoration: none;
        }
        a:hover {
                color: #FF0000; /* red */
                text-decoration: underline;
        }
</style>
        CSS;
}
```

So, once you have your function constructed, it’s time to “hook” the function to the `wp_head` action hook. It’s pretty easy:

```
add_action('wp_head', 'insert_some_css');
```

In layman’s terms, that line of code is saying, “Whenever the `wp_head` action shows up in the theme code, execute the `insert_some_css` function.”

My final code usually looks something like this:

```
add_action('wp_head', 'insert_some_css');
function insert_some_css() {
        echo <<<CSS
<style type="text/css">
<style type="text/css">
        a {
                color: #08FF00; /* green */
                text-decoration: none;
        }
        a:hover {
                color: #FF0000; /* red */
                text-decoration: underline;
        }
</style>
        CSS;
}
```

It’s really not a difficult concept, if you think about it. **All you need to do is find an action hook that you want to use to output some code, create a function, and hook the function to that action using the `add_action` code I gave you above.**

As always, be sure any PHP code is placed between opening and closing PHP tags, or it will not execute.

Creating Your Own Action Hooks

While WordPress does provide (http://codex.wordpress.org/Plugin_API/Action_Reference) you with a LOT of action hooks that you can take advantage of, you can create your own (http://codex.wordpress.org/Function_Reference/do_action) too, **giving your theme or plugin the same benefit of letting other people make changes to your code, without having to edit your code.**

So, if you want to provide an action hook that will execute any functions that are “hooked” to it, just use this code:

```
<?php do_action('my_action_hook_name'); ?>
```

Of course, change the `my_action_hook_name` to something unique that describes the location and/or purpose of the action hook.

And now, anyone can come along and hook their own function to your newly created action hook.

Other Sources on Action Hooks

- Ian Stewart explains action hooks for Child Themes (http://themeshaper.com/action-hooks-wordpress-child-themes/)
- Raymond Selda explains the concept (http://www.raymondselda.com/understanding-action-hooks-in-wordpress/), with examples
- The Codex page for Plugin API (http://codex.wordpress.org/Plugin_API) has some good info
- The Codex lists all the default actions hooks (http://codex.wordpress.org/Plugin_API/Action_Reference) you can use

So, now you know what the fuss is all about. Take it for a spin, and let me know what you think of it in the comments. And if you have any questions, I'd be happy to try to answer them.

Comments

Frank (http://wpengineer.com/) says
June 10, 2009 at 6:50 am (http://www.nathanrice.net/blog/an-introduction-to-wordpress-action-hooks/#comment-1729)

In WordPress it is often better, when you use the function wp_enqueue_style() for add Stylesheet in the head on the site. I think it is important to say this on the example to add css to the head with the hook wp_head.
Thanks for the nice post.

Nathan Rice (http://www.nathanrice.net/) says
June 10, 2009 at 10:27 am (http://www.nathanrice.net/blog/an-introduction-to-wordpress-action-hooks/#comment-1730)

action-hooks/#comment-1730)

Frank,
Very true. If I were adding a call to a stylesheet or JS file, that's what I would have done. But since I was inserting raw styles into the `<head>`, it was better to just hook directly to `wp_head`. The same would have gone for JS code as well.

The enqueue functions are there, primarily, to avoid duplicate calls to the same file, so using them for raw styles or JS has no more benefit than hooking directly to `wp_head`.

Rich (http://www.ridethebeav.com) says
June 10, 2009 at 12:53 pm (http://www.nathanrice.net/blog/an-introduction-to-wordpress-action-hooks/#comment-1732)

I think the hardest thing for people to grasp (me included) is how to change the layout of a parent theme using action hooks. Someone (who I am not going to mention) said that most child themes look pretty much exactly like the parent theme. The reason for this is that changing the actual structure (not the link color) is a bit more advanced topic, and one that has yet to be addressed thoroughly.

What I would like to see as a followup is a literal example. Take one of the two major frameworks (thesis or thematic) and completely change the layout to something that could no longer resemble the parent theme. When people realize how flexible these frameworks are, I think people might have their ah ha moment. Most of us are still like, "ok but now what"

Thanks for the intro though, I always enjoy your articles.

Rich

Nathan Rice (http://www.nathanrice.net/) says
June 10, 2009 at 1:09 pm (http://www.nathanrice.net/blog/an-introduction-to-wordpress-action-hooks/#comment-1733)

That’s a pretty good idea. I’ve got a theme framework that I’ve been working on with over 32 action hooks that I’ll be releasing for beta testing soon. Once that happens, I’ll do plenty of examples on how to change the default structure by using action hooks. I would do it with the other 2 frameworks you mentioned, but I simply don’t know enough about them to do them justice.

Rich (http://www.ridethebeav.com) says
June 10, 2009 at 1:13 pm (http://www.nathanrice.net/blog/an-introduction-to-wordpress-action-hooks/#comment-1734)

I pretty much figured you’d have something in the works, Nathan. I will be impatiently waiting for this.

Rich

Nathan Rice (http://www.nathanrice.net/) says
June 11, 2009 at 1:24 pm (http://www.nathanrice.net/blog/an-introduction-to-wordpress-action-hooks/#comment-1739)

I’ll try not to make you wait too long. I want to roll out a developer beta as soon as I can. But I need to finish up the default child theme, and set up a forum to discuss development with everyone, so that we can get it ready for a public release.

j.verhine (http://www.verhine.com) says
June 11, 2009 at 12:07 am (http://www.nathanrice.net/blog/an-introduction-to-wordpress-action-hooks/#comment-1736)

assuming that you were creating an action hook for a theme, where would you place the function?

Nathan Rice (http://www.nathanrice.net/) says
June 11, 2009 at 1:23 pm (http://www.nathanrice.net/blog/an-introduction-to-wordpress-action-hooks/#comment-1738)

The function could be created in your theme's functions.php file, or in a separate plugin. The Action Hook (the `do_action()` thing) would need to be placed where you want the action hook to execute, wherever that might be.

Raymond Selda (http://www.raymondselda.com/) says
June 15, 2009 at 5:14 am (http://www.nathanrice.net/blog/an-introduction-to-wordpress-action-hooks/#comment-1741)

Nice article Nathan. Thank you for including my article as a resource. Looking forward to your theme framework. I'm also developing my own framework with bits and pieces of Thematic's source and I hope to release it prematurely as soon as possible. hahaha. Thanks again.

Brent Shepherd (http://find.brentshepherd.com) says
July 9, 2009 at 9:21 pm (http://www.nathanrice.net/blog/an-introduction-to-wordpress-action-hooks/#comment-1746)

Ah wow, thank you for this article!

The paragraph about creating your own action hooks was particularly useful.

After reading the WP documentations description of do_action, I didn't fully comprehend this. "Creates a hook for attaching actions via add_action." just isn't as clear as your description.

Do you mind if I paraphrase your explanation for the documentation?

shane says
July 23, 2009 at 4:23 pm (http://www.nathanrice.net/blog/an-introduction-to-wordpress-action-hooks/#comment-1749)

Thanks for this. I've been trying to learn about these for a little while now and believe it or not, guides like yours seem to be few in number.

filipe says
September 7, 2009 at 1:27 am (http://www.nathanrice.net/blog/an-introduction-to-wordpress-action-hooks/#comment-1753)

THANKS A LOT!! finally a I understood the code and put a link in footer throut function !

TY again !!

Christine Green (http://www.christinegreen.com/) says
October 24, 2009 at 8:01 pm (http://www.nathanrice.net/blog/an-introduction-to-wordpress-action-hooks/#comment-1754)

Thanks for the intro to Action Hooks. It's all getting clear now.

Hikari (http://tutorial.Hikari.ws/) says
December 26, 2009 at 6:17 pm (http://www.nathanrice.net/blog/an-introduction-to-wordpress-action-hooks/#comment-1757)

Another good advantage of using hooks is that you can replace original code with yours.

In that way, when you are developing some code, you store it in a function and just below hooks this function with `add_action` and just after you call `do_action`.

A better description for `do_action` is that it calls all functions that are hooked to it, as if it was a list of functions calls. Doing as I said, your function will be called as if you had called it there.

But the difference of calling a function and using an action hook, aside from the action allowing

But the difference of calling a function and using an action hook, aside from the action allowing to call a bunch of functions at once and letting other ppl to call their functions inside your code, is that other ppl can use the **`remove_action`** function to **remove** your original function from being called. And after it, they can develop their own function and add it there, **replacing** your original function from their, which may be better than yours or just do something differently that fits better their needs.

That's a tremendous powerful feature!!!

And since you are teaching ppl how to do it, I think it is also important to teach them the importance of using prefixes on our functions (and hooks too!), so that they don't conflict with others'.