



The Handbook

Codex

http://codex.wordpress.org/I_Make_Changes_and_Nothing_Happens

Version Date

6 August 2005

I Make Changes and Nothing Happens

Things are going great. You have figured out how to [write a post](#), how to [make a few categories](#), and maybe even [add a plugin](#) or two. Then IT HAPPENS. You make a few changes in your WordPress Theme and when you view your WordPress site, *nothing has changed*. Your fix isn't fixed. Your change isn't changed. NOTHING HAS CHANGED! You want to scream, pull out your hair, pound on your computer, and shout blame.

Hang on. **Calm down**. A great many errors are in fact mistakes born of haste. If you're not operating under a tight deadline, try to **remain calm, and collected**, as you proceed. It will help. If you start to get unduly tense, or frustrated, simply get up and walk away for a few minutes. Sometimes all it takes is a fresh, rested pair of eyes to spot a missing semi-colon, or an extra quote mark, or to realize that the solution is much simpler than you originally thought.

The problem might be with WordPress, it might be with your database, it might be with your server, or it might be something you did to screw things up, but the reality is that a lot of the time it is none of those things. The culprit is probably your Internet browser.

Whatever the problem is, we're here to help. Let's look at the possible problems and solutions for what to do when you make a change and nothing happens.

The Browser Cache

Did you know that a computer is supposed to make your life easier? Less complicated? It is supposed to save you time and energy and actually improve your life. No? Well, maybe not, but your Internet browser does its best to try to make your life a little easier.

When you first visit a web page, it often takes a while to load, right? But the next page you visit within that site doesn't take so long to load. This is because, in an effort to be helpful, the browser stores the information on your computer so it reloads it from your computer, not from the actual site. This is called the **cache** and it is meant to speed up your Internet browsing.

The term *cache* may sound familiar. Remember the pirates and thieves of old who would stockpile their treasures in a cave, hole in the ground, or somewhere "safe". Called the *cache*, the Internet browser stocks away files and information for the browser to reuse when the page is refreshed or viewed again.

The problem comes when you make a small change to your site and the browser doesn't recognize it as a significant change, so it reloads the same thing you just looked at. The solution is to clear or empty your **browser's cache**.

Clearing the Browser Cache

Normally, to see the changes on your page, you click the **Refresh** button on the browser toolbar or the F5 key on your keyboard. In many cases, this simply reloads the page without clearing the browser's cache. Here are some techniques to totally clear the browser's cache so you will see the changes when your page reloads.

Microsoft Internet Explorer

1. Hold down SHIFT and click on the REFRESH button in the toolbar under the menu.
2. **For Serious Clearing:** If you are having problems clearing out the cache, then force it by choosing from the menu TOOLS > INTERNET OPTIONS > TEMPORARY FILES. Click on **Delete Temporary Files**. You can choose the checkbox to delete **all** Internet files, but you might not want to as that will also clear all your passwords and cookies, but if you are having trouble viewing the changes on your page, go all the way.

FireFox

1. Hold down CTRL+SHIFT+R.
2. Using the FireFox Web Developer Extension (add-on), click Miscellaneous and then Clear Cache.
3. **For Serious Clearing:** From the browser menu, Tools > Options > Privacy > Cache and select Clear.

Netscape

From the menu, click Edit > Preferences > Advanced. Choose "Cache" and click both "Clear Memory Cache" and "Clear Disk Cache".

Mozilla 1.x and up

From the browser menu, Edit > Preferences > Advanced and click "Cache" and "Clear Cache".

Opera

From the browser menu, Edit > File > Preferences > History and Cache and click "Cache".

Safari

From the browser menu, Safari > Reset Safari and click Reset to confirm OR Safari > Empty Cache.

Set Your Browser to Not Cache

Each browser may have a way of stopping or minimizing the caching of web pages. Using this technique will definitely slow down your web page viewing, and it isn't a perfect solution because some caching may still occur, but it does help. Check your Internet browser's help files for the specifics on how to turn off the cache feature.

Check Your Source

You know, even the very best web page designers, developers, and programmers screw up. It's the little details, the forgotten semi-colon, the misspelled tag, the lack of attention to a detail that screws things up. If the best do it, then it's very possible you have overlooked a little detail. And if you did, well, welcome to the club. It's a part of the process. Let's look at some of the most commonly overlooked details that happen when you aren't paying attention.

Check the Address

Is the name and folder for the file you "fixed" the same as the one you are viewing? Look at the following two addresses (URLs).

- `wordpress/wp-content/themes/yourtheme/style.css`
- `test/wordpress/wp-content/themes/yourtheme/style.css`

In this case, you can probably see the difference, but when viewed in an address bar or in a text editor, you might miss the word `test` that sets the folder.

Pay very close attention to the difference between `style1.css` and `stylel.css` if you are using different style names, too. The first filename is `style` followed by the digit one, while the second filename is `style` followed by a lowercase L. If you are working with different but similar files, make sure you give them distinctive names like `style-red.css` and `style-800.css` so you can clearly see the difference.

Check Your Upload

When you make a change in a file, it is often on your computer's hard drive and you have to upload the file to your host server in order to view it on the Internet. Did you actually upload it? Did you put it in the right folder? Is it really there? When over-writing the exact same file, it doesn't always do a complete over-write, so consider deleting the original on the host server and then uploading the new version to make sure the right and whole thing is there.

Test Yourself

If you still can't see the changes you made, and the file is in the right place with the right name, and you are sure it's the right file, then go through these steps:

1. Make a backup of the file you are working on and check that the backup is in a safe place.
2. Make a big change (such as setting the background in your `style.css` as `#ff000` or even red).
3. View the changed web page in your browser. Make sure you clear the cache to be sure you have the new version.
4. If nothing changes, delete the file (and only that file) from the server and try to view the file again. If nothing continues to change, you and WordPress are looking at completely different files. It's time to get out your detective hat and start tracking down what is going on and where your files went to.
5. Check your URL settings in your Options Panel and also in the database, and if this continues to be unsolvable, post a note explaining what you've done and what's the result on the [WordPress Forum](http://www.wordpress.com/support) (<http://www.wordpress.com/support>) and let the experts step in to help.

Debug It

Programmers use the term **debug** to describe the process of going through code and finding the little criminal that is messing things up. We're going to look at how to debug your CSS, HTML, and PHP code to help you figure out why you can't see the changes you have made.

When debugging a problem, **change only one thing at a time**. If you're not sure if the problem is in line 37 or line 40, don't change both lines in one go! First change line 37, and see if that fixed the problem. If not, *undo* the change you made on line 37 and then make a change on line 40. It's important to *undo* proposed fixes before trying something else, even

if the first attempt doesn't immediately introduce new problems.

Every time you make a change to a file, you run the risk of accidentally making more mistakes. These sorts of things tend to cascade, making debugging an infuriating process. Do one thing at a time.

Debugging CSS

Debugging your style sheet, or your CSS, can be challenging because you have to find the area in the HTML that is causing the trouble and then track that section back to its style in your `style.css` file. The article on [Finding Your CSS Styles](#) will help you find those troublesome sections.

Once you have found the style that is causing the problem, you need to figure out what it is about the style that is causing the problem. Here is a quick checklist of things to consider as you troubleshoot your CSS:

- Is everything spelled right?
- Is every period, brace, colon and semi-colon in its right place?
- Are you using the style attribute or *declaration* correctly?
- Is there a declaration that shouldn't be in there, like `font-weight:x-large`? The declaration `x-large` is used for `font-size` not `font-weight`.
- Do you have spaces in places where they shouldn't be, like inside of a `background-image:url (' bg.gif ')`? (correct: `background-image:url('bg.gif')`)

You can find more information to help you debug and troubleshoot your CSS with:

- [CSS Troubleshooting](#)
- [Finding Your CSS Styles](#)
- [CSS Fixing Browser Bugs](#)
- Troubleshooting Your Theme - *Coming soon*

Debugging HTML

Similar to debugging CSS, HTML can also get bogged down with careless little mistakes like misspellings, forgotten closing tags, forgotten `<` arrows, and other little errors that can send your site into twisted remains.

It is highly recommended that you use one of the [HTML Validators](#) available for free on the Internet. Also, [Mozilla Firefox Internet Browser](#) (<http://www.mozilla.org/products/firefox/>) has a powerful free add-on called the [Web Developers DOM Inspector](#) (<http://www.mozilla.org/projects/inspector/>) that will help you validate and fix your website problems very quickly and easily.

Some tips for debugging your HTML/XHTML may include:

- Improperly nested XHTML, especially in [nested lists](#) commonly found in the sidebar.
- Unclosed tags.
- Self-closing tags not closed by use of the forward slash (example: ``).
- Incorrect tag usage.

For more help on debugging your HTML, check out these articles and resources:

- [Validating a Website](#)
- [Stepping Into Template Tags](#)

Debugging PHP

If you have access to your webserver's error log, take time to check it. PHP usually logs informative errors here when things go wrong. These log messages can sometimes be a little cryptic, but they should always give the line number of the offending piece of code.

Unfortunately, what PHP thinks is the offending piece of code is sometimes not the problem. For example, an unclosed curly brace `{` may be reported as a problem on some line number way on down in your script. A quick cheat sheet of PHP error messages and their common causes can be found at [Common PHP Errors](#).

To debug your PHP, here are the steps you should follow:

See Where You Are

The `die()` command is probably the single most useful debugging tool available. `die()` immediately halts execution of the program, optionally displaying a message of your choosing.

When trying to work through problems, sprinkle `die()` statements in at key sections of your script, giving each an informative message:

```
die('Stage One Complete');  
die('Disinfribullation Complete');  
die('Finished Collating');
```

Load your program, and see which message (if any) gets displayed. If you see the first message, you know your program got that far without error. You can safely remove (or comment out) that `die()` command, and re-run your program, to see how much farther it got. Step through your program this way until you've isolated the problem area.

This technique has some limitations, though. First, if **nothing** is being displayed in your browser, then most likely you have a fatal syntax error somewhere in your script. Check your webserver's error log, if possible. If the script executes completely -- but incorrectly -- and none of your `die()` messages are displayed, then you've got some more work to do.

See What's What

If things aren't being set as you want them, or stuff isn't happening when you want, you'll likely need to check the value of different variables at different places of your script. Simply pass the variable to a `die()` command, and you'll be able to see what that variable's value is:

```
die($user_level);
```

This will display the value of the variable `$user_level`, which should reveal the current user's user level.

This technique is fine for some variable types, like boolean and string

("scalar (<http://www.php.net/manual/en/language.types.php#language.types.intro>)") variables in programmer parlance; but fails for [arrays](http://www.php.net/manual/en/language.types.array.php) (<http://www.php.net/manual/en/language.types.array.php>) and [objects](http://www.php.net/manual/en/language.types.object.php) (<http://www.php.net/manual/en/language.types.object.php>). To see the value(s) of arrays and objects, use the [print_r](http://www.php.net/print_r) (http://www.php.net/print_r) command:

```
print_r($post);
```

It will display the value of the \$post array in a human-readable format:

```
Array ( [0] => stdClass Object ( [ID] => 1 [post_author]
=> 1 [post_date] => 2005-02-16 09:16:46 [post_date_gmt]
=> 2005-02-16 14:16:46 [post_content] => Welcome to
WordPress. This is your first post. Edit or delete it,
then start blogging! [post_title] => Hello world! [post_category]
=> 0 [post_excerpt] => [post_status] => publish [comment_status]
=> open [ping_status] => open [post_password] => [post_name]
=> hello-world [to_ping] => [pinged] => [post_modified] => 2005-04-15
08:45:42 [post_modified_gmt] => 2005-04-15 13:45:42
[post_content_filtered] => [post_parent] => 0 [guid]
=> /?p=1 [menu_order] => 0 ) )
```

This format allows you to see what each key/value pair is inside the array.

Using `print_r()` does not stop execution of the program, so it is often necessary to call `die()` immediately afterwards.

Use `print_r()` and `die()` to check the values of your variables through the execution of your script. There's also a `var_dump()` function which works similarly. Use whichever mechanism you find most informative.

When All Else Fails

If all else fails, know this: **You are not alone.** We've all been there. To help you get back on track and your site up and running again, check the following resources for more help:

- [Troubleshooting](#)
- [WordPress Lessons](#)