10 things you (probably) didn't know about PHP.

PHP is simultaneously the most infuriating and joyful languages you've probably ever worked with. "infuriating" primarily because the function names are so inconsistant. Here's a short list of cool features that might have slipped under your radar.

PHP is simultaneously the most infuriating and joyful languages I've ever worked with. I say "infuriating" primarily because the function names are so inconsistant. Despite the fact that I use it almost everyday, I still have to think to myself "Is it *str_pos* or *strpos*? *str_split* or *strsplit*?" On the other hand, occasionally I'll stumble across a gem that perfectly solves the problem at hand with a single line of code.

Here's a short list of cool features that might have slipped under your radar as well:

1. **Use *ip2long()* and *long2ip()* to store IP addresses as integers instead of strings in a database.** This will reduce the storage space by almost a factor of four (15 bytes for char(15) vs. 4 bytes for the integer), make it easier to calculate whether a certain address falls within a range, and speed-up searches and sorts (sometimes by quite a bit).
2. **Partially validate email addresses by checking that the domain name exists with *checkdnsrr()*.** This built-in function checks to ensure that a specified domain name resolves to an IP address. A simple user-defined function that builds on checkdnsrr() to partially valid email addresses can be found in the user comments section in the PHP docs. This is handy for catching those occasional folks who think their email address is 'joeuser@wwwphp.net' instead of 'joeuser@php.net'.
3. **If you're using PHP 5 with MySQL 4.1 or above, consider ditching the *mysql_** functions for the improved *mysqli_** functions.** One nice feature is that you can use prepared statements, which may speed up queries if you maintain a database-intensive website. Some benchmarks.
4. **Learn to love the ternary operator**.
5. **If you get the feeling that you might be reinventing the wheel during a project, check *PEAR* before you write another line.** PEAR is a great resource that many PHP developers are aware of, yet many more are not. It's an online repository containing over 400 reusable snippets that can be dropped right into your PHP application. Unless your project is trully unique, you ought to be able to find a PEAR package that saves at least a little time. (Also see PECL)
6. **Automatically print a nicely formatted copy of a page's source code with highlight_file().**This function is handy for when you need to ask for some assistance with a script in a messageboard, IRC, etc. Obviously, some care must be taken not to accidently show your source when it contains DB connection information, passwords, etc.
7. **Prevent potentially sensitive error messages from being shown to users with the *error_reporting(0)* function.** Ideally error reporting should be completely disabled on a production server from within *php.ini*. However if you're on a shared webhost and you aren't given your own *php.ini*, then your best bet is to add *error_reporting(0);* as the first line in each of your scripts (or use it with require_once().) This will prevent potentially sensitive SQL queries and path names from being displayed if things go awry.
8. **Use *gzcompress()* and *gzuncompress()* to transparently compress/decompress large strings before storing them in a database.** These built-in functions use the gzip algorithm and can compress plaintext up to 90%. I use these functions almost everytime I read/write to a BLOB field within PHP. The only exception is when I need full text indexing capabilities.
9. **Return multiple values from a function with "by reference" parameters.** Like the ternary operator, most PHP developers who come from a more formalized programming background already know this one. However, those who's background is more HTML than Pascal, probably have wondered at one time "how do I get multiple values back from a function I wrote, even though I can only use one *return* value?" The answer is that you precede a variable with "&" and use it "by reference" instead of "by value".
10. **Fully understand "magic quotes" and the dangers of SQL injection.** I'm hoping that most developers reading this are already familiar with SQL injection. However, I list it here because it's absolutely critical to

understand. If you've never heard the term before, spend the entire rest of the day googling and reading.
