Hidden Features of PHP? [closed]

Asked 16 years, 3 months ago Modified 12 years, 1 month ago
Viewed 64k times Part of PHP Collective

174
votes

It's difficult to tell what is being asked here. This question is ambiguous, vague, incomplete, overly broad, or rhetorical and cannot be reasonably answered in its current form. For help clarifying this question so that it can be reopened, visit the help center.

Closed 14 years ago.

Locked. This question and its answers are locked because the question is off-topic but has historical significance. It is not currently accepting new answers or interactions.

I know this sounds like a point-whoring question but let me explain where I'm coming from.

Out of college I got a job at a PHP shop. I worked there for a year and a half and thought that I had learned all there was to learn about programming.

Then I got a job as a one-man internal development shop at a sizable corporation where all the work was in C#. In my commitment to the position I started reading a ton of blogs and books and quickly realized how wrong I was to think I knew everything. I learned about unit testing, dependency injection and decorator patterns, the design principle of loose coupling, the composition over inheritance debate, and so on and on and on - I am still very much absorbing it all. Needless to say my programming style has changed entirely in the last year.

Now I find myself picking up a php project doing some coding for a friend's start-up and I feel completely constrained as opposed to programming in C#. It really bothers me that all variables at a class scope have to be referred to by appending '$this->' . It annoys me that none of the IDEs that I've tried have very good intellisense and that my SimpleTest unit tests methods have to start with the word 'test'. It drives me crazy that dynamic typing keeps me from specifying implicitly which parameter type a method expects, and that you have to write a switch statement to do method overloads. I can't stand that you can't have nested namespaces and have to use the :: operator to call the base class's constructor.

Now I have no intention of starting a PHP vs C# debate, rather what I mean to say is that I'm sure there are some PHP features that I either don't know about or know about yet fail to use properly. I am set in my C# universe and having trouble seeing outside the glass bowl.

So I'm asking, what are your favorite features of PHP? What are things you can do in it that you can't or are more difficult in the .Net languages?

PHP php hidden-features

Share

edited Nov 16, 2012 at 17:08

community wiki
9 revs, 4 users 54%
George Mauer

Broken OO paradigm? For me it's the worst "hidden" feature you discover. – knoopx Dec 26, 2009 at 18:25

These threads are kind of funny... Because for the team I work with "hidden feature" is a code phrase meaning "bug". And you know what... Sometimes discovering a hidden feature is not necessarily a good thing... – Ganesh Shankar Feb 23, 2010 at 23:44

@Ganesh *one man's bug is another mans hidden feature*... – Xeoncross Jun 28, 2010 at 23:03

Comments disabled on deleted / locked posts / reviews

78 Answers

Sorted by: Highest score (default)

1 2 3 Next

328 votes

Documentation. The documentation gets my vote. I haven't encountered a more thorough online documentation for a programming language - everything else I have to piece together from various websites and man pages.

Share

edited Feb 23, 2010 at 20:28

community wiki
2 revs, 2 users 86%
Kyle Cronin

60 I agree. Being able to type www.php.net/function_name and getting a reference most of the time is great. – Allain Lalonde Sep 14, 2008 at 17:46

1 This is a great feature of PHP, but I wouldn't really call it hidden... If you've ever googled for method parameters, etc, you would end up at php.net. – John B ♦ Mar 23, 2009 at 17:04

27 I agree as well. The greatest thing about the manual are the user comments. I have rarely seen other documentations have those. They can contain real gems. The only downside is that IMHO they a pruned little too soon. – Sander Marechal Jun 21, 2009 at 22:37

3 @Phoexo "little bit less read-able" ??? I never understanded and will never understand MSDN, while PHP docs are easy and clear. – lauriys Feb 13, 2010 at 12:13

3 Disagree. The only reason the documentation is "good" is because of some of the user comments plugging all the holes in the official notes. – Rob Howard Feb 27, 2011 at 8:28

179 votes

Arrays. Judging from the answers to this question I don't think people fully appreciate just how easy and useful Arrays in PHP are. PHP Arrays act as lists, maps, stacks and generic data structures all at the same time. Arrays are implemented in the language core and are used all over the place which results in good CPU cache locality. Perl and Python both use separate language constructs for lists and maps resulting in more copying and potentially confusing transformations.

Share

edited Feb 23, 2010 at 20:26

community wiki
2 revs, 2 users 67%
user8134

11 PHP array elements are ordered. – user8134 Feb 17, 2009 at 19:02

117 My initial move from PHP to C# almost killed me. In C#, arrays are just a simple structure with a static size and numeric index. In PHP, arrays are the duct tape of the universe! – Dinah Mar 25, 2009 at 16:04

4 I also agree. When playing with Java for a Uni assignment, I was stunned at how rigid they were, no flexibility at all. Really made me appreciate just how good PHP arrays are. – Christian Jun 22, 2009 at 0:54

11 I'm sure php arrays are great, but 40 votes for the comment that knocks C# arrays? If a C# array doesn't fit the need, there are many options. ArrayList and generic collections are both very powerful. There are numerous types of collections for specific needs. The only actual *advantage* of php in this regard is that it doesn't provide any options from which a programmer must decide. You either use array or you don't have an indexable variable. – G-Wiz Jan 9, 2010 at 7:01

24 On the other hand, the syntax for arrays completely sucks. In many scripting languages you can create a simple 2D array like so: `[[1, 2], [3, 4]]`, compare this to the PHP version: `array(array(1, 2), array(3, 4))`.
– Rene Saarsoo Feb 4, 2010 at 8:43

167
votes

Stream Handlers allow you to extend the "FileSystem" with logic that as far as I know is quite difficult to do in most other languages.

For example with the MS-Excel Stream handler you can create a MS Excel file in the following way:

```
$fp = fopen("xlsfile://tmp/test.xls", "wb");
if (!is_resource($fp)) {
    die("Cannot open excel file");
}

$data= array(
    array("Name" => "Bob Loblaw", "Age" => 50),
    array("Name" => "Popo Jijo", "Age" => 75),
    array("Name" => "Tiny Tim", "Age" => 90)
);

fwrite($fp, serialize($data));
fclose($fp);
```

Share

edited May 2, 2010 at 19:48

community wiki
3 revs
Allain Lalonde

I believe that the KIO framework lets you do this as well, but that's only available for KDE-based desktop applications. – MiffTheFox Jun 6, 2009 at 7:26

21 IMHO having a proper OO approach would be much more sensible than this mess with stream handlers. Yes, its cute to be able to read/write Excel files, but does it have to work like this? – Anti Veeranna Aug 22, 2009 at 20:50

3 Maybe so, but this approach encapsulates complexity in an interface that's common to most PHP developers... without requiring them to learn Object Oriented concepts which might be beyond them. – Allain Lalonde Aug 23, 2009 at 13:59

13 If you're working with Amazon S3, check out Zend_Amazon_S3, which provides a stream interface for urls like 's3://{bucket-name}/path'. – davidtbernal Sep 13, 2009 at 0:21

I've used this to create a simple DSL for my view layer by just reading the PHP file, doing some string replacement and passing it trough eval(). Eg, I made it such that I can use short-tags whenever I choose to and do @->someVar so I can access view-level data. – Jasper Bekkers Dec 10, 2009 at 11:21

131 votes

Magic Methods are fall-through methods that get called whenever you invoke a method that doesn't exist or assign or read a property that doesn't exist, among other things.

```
interface AllMagicMethods {
    // accessing undefined or invisible (e.g. private)
    public function __get($fieldName);
    public function __set($fieldName, $value);
    public function __isset($fieldName);
    public function __unset($fieldName);
    // calling undefined or invisible (e.g. private) me
    public function __call($funcName, $args);
    public static function __callStatic($funcName, $arg

    // on serialize() / unserialize()
    public function __sleep();
    public function __wakeup();

    // conversion to string (e.g. with (string) $obj, e
...)
    public function __toString();

    // calling the object like a function (e.g. $obj($a
    public function __invoke($arguments, $...);

    // called on var_export()
    public static function __set_state($array);
}
```

A C++ developer here might notice, that PHP allows overloading some operators, e.g. `()` or `(string)`. Actually PHP allows overloading even more, for example the `[]` operator (ArrayAccess), the `foreach` language construct (Iterator and IteratorAggregate) and the `count` function (Countable).

Share

edited Aug 28, 2010 at 8:58

community wiki
6 revs, 4 users 64%
NikiC

4 As useful example of what can be achieved with magic methods goto phpcodetips.blogspot.com/2008/07/domain-model-validation.html – grom Oct 28, 2008 at 23:44

6 Disagree. This is far weaker than similar facilities in Smalltalk, Ruby & Python (and presumably it was copied from one of these) – finnw Jun 21, 2009 at 21:49

34 The fact that PHP's implementation of this functionality is weaker than those other languages, doesn't make it any less useful in PHP. – Allain Lalonde Jun 22, 2009 at 2:07

2 `__call()` is great in frameworks with map domain.com/controller/method/ – alex Jul 9, 2009 at 12:30

7 Magic methods are also slow as hell. Use them carefully. – Alex Weinstein Jul 29, 2009 at 20:07

95
votes

The **standard class** is a neat container. I only learned about it recently.

Instead of using an array to hold serveral attributes

```
$person = array();
$person['name'] = 'bob';
$person['age'] = 5;
```

You can use a standard class

```
$person = new stdClass();
$person->name = 'bob';
$person->age = 5;
```

This is particularly helpful when accessing these variables in a string

```
$string = $person['name'] . ' is ' . $person['age'] . '
// vs
$string = "$person->name is $person->age years old.";
```

Share

edited Jul 9, 2009 at 15:17

community wiki
4 revs, 2 users 97%
Dean Rather

43 "{$person['name']} is {$person['age']} years old" works. – Kornel Nov 16, 2008 at 23:56

27 "person[name] is $person[age] years old" will also work... No quotes, no braces :) – majelbstoat Nov 24, 2008 at 1:44

16 $string = sprintf("%s is %d years old.", $person['name'], $person['age']); – Daniel Sloof May 10, 2009 at 2:00

60 While we're on the subject: (object)array("name" => 'bob', 'age' => 5) – user7675 Jun 6, 2009 at 21:46

30 @majelbstoat: Taking out the quotes would slow the script down because the PHP interpreter will look to see if 'name' and 'age' have been set with define(...). It's also a bad practice considering it'd be possible to totally flip the keys that are accessed in each case: define('age','name'); define('name','age'); – brianreavis Sep 3, 2009 at 3:59

votes

Include files can have a return value you can assign to a variable.

```
// config.php
return array(
    'db' => array(
        'host' => 'example.org',
        'user' => 'usr',
        // ...
    ),
    // ...
);

// index.php
$config = include 'config.php';
echo $config['db']['host']; // example.org
```

Share

edited Feb 28, 2011 at 23:56

community wiki
4 revs, 3 users 94%
Philippe Gerber

@Peter *VERY* useful for db->localhost exception error handling. – Talvi Watia Jan 29, 2010 at 11:30

It's convenient for setting up a quick-and-dirty config file. – Frank Farmer Feb 23, 2010 at 21:27

Why do you return this array? If an included file contains an array it's usable at the include immediately. – fabrik Aug 11, 2010 at 7:47

5 @fabrik because it would be global variable and available in the whole main scope. That's quite unpleasant, this is way better. – Mikulas Dite Aug 28, 2010 at 9:14

i worked on a project in Yii framework and that project have config file in which array was returned like this, now i understand why file was like this.
– Simer Twilio Toronto developer Dec 23, 2010 at 12:01

83
votes

You can take advantage of the fact that the `or` operator has lower precedence than `=` to do this:

```
$page = (int) @$_GET['page']
  or $page = 1;
```

If the value of the first assignment evaluates to `true`, the second assignment is ignored. Another example:

```
$record = get_record($id)
  or throw new Exception("...");
```

Share

edited Mar 12, 2009 at 14:14

community wiki
6 revs
Pies

7 I'm not quite convinced of this I think; even though it seems not to be error prone it's counter-intuitive, and that in itself may promote errors. – thomasrutter Mar 20, 2009 at 10:01

14 @Pies: one way is the following, quite messy code to be honest: $page = isset($_GET['page']) ? (int)$_GET['page'] : 1; // Advantage of this is no error suppression is required. – DisgruntledGoat May 9, 2009 at 22:00

3 on second thoughts, since you're looking for an integer you could use instead: $page = is_int($_GET['page']) ? $_GET['page'] : 1; – DisgruntledGoat May 9, 2009 at 22:13

4 It's worth noting that the code after `or` will execute if the code before `or` results in the numeric value `0`. So semantically it may be less likely with something like `$_GET['page']`, but obviously the circumstance may arise and it's good to watch out for. – eyelidlessness Dec 10, 2009 at 5:25

3 It's also worth noting that the `or` operator is a lower-precedent version of the `||` operator. Also, +1 because this is highly expressive and I often forget it's possible. It should be used more often, and it's absolutely clear in what it does. I don't know about how "real males" code though, so I can't comment on that. – eyelidlessness Dec 10, 2009 at 5:27

80 votes

`__autoload()` (class-) files aided by `set_include_path()`.

In PHP5 it is now unnecessary to specify long lists of "include_once" statements when doing decent OOP.

Just define a small set of directory in which class-library files are sanely structured, and set the auto include path:

```
set_include_path(get_include_path() . PATH_SEPARATOR . '
```

Now the `__autoload()` routine:

```
function __autoload($classname) {
    // every class is stored in a file "libs/classname.c

    // note: temporary alter error_reporting to prevent
    // Do not suppress errors with a @ - syntax errors w

    include_once($classname . '.class.php');
}
```

Now PHP will automagically include the needed files on-demand, conserving parsing time and memory.

Share

edited Feb 28, 2011 at 14:04

community wiki
6 revs, 5 users 76%
Wimmer

is spl_autoload_register() better to use? – alex Apr 21, 2009 at 2:04

19 Of course! __autoload() is PHP4 but spl_autoload_register() is a non-destructive "daisy-chaining" of autoloading methods. – Willem Apr 21, 2009 at 20:00

3 A handy feature, but the one caveat is when you find an instance of a given class, it makes it a little more difficult to hunt down the location of a class file. Explicitly defining includes at the top gives you a finite list of involved classes and their exact location. – Cory House Apr 25, 2009 at 3:20

It's a nice feature, but only to get around the situation of not having code precompiled, so it doesn't know where the class is going to be. Big downside of this is that you can't have 2 classes with the same name, even if they are in different namespaces. – Kibbee Jul 1, 2009 at 1:59

3 Please have a look at the PSR-0 propsal from the PHP Standards Working Group (featuring developers of ZF, Symfony, Doctrine, CakePHP, Solar, etc.) when implementing autoloading: groups.google.com/group/php-standards/web/psr-0-final-proposal – Philippe Gerber Aug 29, 2010 at 11:18

76 votes

Easiness. The greatest feature is how easy it is for new developers to sit down and write "working" scripts and understand the code.

The worst feature is how easy it is for new developers to sit down and write "working" scripts and think they understand the code.

The **openness of the community** surrounding PHP and the massive amounts of PHP projects available as open-source is a lot less intimidating for someone entering the development world and like you, can be a stepping stone into more mature languages.

I won't debate the technical things as many before me have but if you look at PHP as a community rather than a web language, a community that clearly embraced you when you started developing, the benefits really speak for themselves.

Share edited Feb 23, 2010 at 20:35

community wiki
2 revs, 2 users 86%
Chris Ridenour

3 Definitely a good comment. Python was my first language and it was awesome, but the lack of projects I felt capable of understanding did create a barrier. With PHP I can look up pretty much anything on the documentation and figure it out...the resources available on the web are amazing. – dscher Apr 6, 2010 at 3:49

76
votes

Variable variables and functions without a doubt!

```
$foo = 'bar';
$bar = 'foobar';
echo $$foo;    //This outputs foobar

function bar() {
    echo 'Hello world!';
}

function foobar() {
    echo 'What a wonderful world!';
}
$foo();    //This outputs Hello world!
$$foo();    //This outputs What a wonderful world!
```

The same concept applies to object parameters ($some_object->$some_variable);

Very, very nice. Make's coding with loops and patterns very easy, and it's faster and more under control than eval (Thanx @Ross & @Joshi Spawnbrood!).t

Share

edited Feb 23, 2010 at 20:36

community wiki
6 revs, 2 users 97%
Jrgns

111 variable variables are actually making the code less readable and more prone to errors. – Elzo Valugi Jul 9, 2009 at 12:30

8 And people really use this? Man i'd hate to read these sources. – Gary Willoughby Jul 25, 2009 at 23:29

27 Variable variables are one of the worst features PHP offers. – davidtbernal Sep 13, 2009 at 0:22

10 9 Out of 10 times variable variables are better replaced with arrays so you have all the data in one place, you can iterate over it et cetera. Only in some very specific circumstances might they be useful. – Jasper Bekkers Dec 10, 2009 at 11:24

7 Please dont make newbies use that "feature". – whiskeysierra Dec 21, 2009 at 13:29

68 votes

You can use **functions with a undefined number of arguments** using the `func_get_args()`.

```
<?php

function test() {

    $args = func_get_args();
    echo $args[2]; // will print 'd'
    echo $args[1]; // will print 3
}
```

```
test(1,3,'d',4);

?>
```

Share

edited Feb 23, 2010 at 20:48

community wiki
4 revs, 4 users 70%
TheBrain

1 Was just about to post this. Sort of like the arguments property in JS functions. – alex Jul 9, 2009 at 12:49

67
votes

I love **remote files**. For web development, this kind of feature is exceptionally useful.

Need to work with the contents of a web page? A simple

```
$fp = fopen('http://example.com');
```

and you've got a file handle ready to go, just like any other normal file.

Or how about reading a remote file or web page directly in to a string?

```
$str = file_get_contents('http://example.com/file');
```

The usefulness of this particular method is hard to overstate.

Want to analyze a remote image? How about doing it via FTP?

```
$imageInfo =
getimagesize('ftp://user:password@ftp.example.com/image/
```

Almost any PHP function that works with files can work with a remote file. You can even `include()` or `require()` code files remotely this way.

Share

edited Feb 23, 2010 at 21:17

community wiki
2 revs, 2 users 98%
zombat

2 This is *so* nice! To do this in for example Java you need to include a gazillion jar files and than write a lot of boilerplate code. – Kimble Dec 26, 2009 at 18:34

16 "You can even include() or require() code files remotely this way." Of course, include()ing a file on a server you don't control is a terrible, terrible idea. – Frank Farmer Feb 23, 2010 at 21:22

4 @Frank - yes, well, one would presume that you would be including code from a server you **did** control. – zombat Feb 24, 2010 at 23:23

1 Point being, remote file inclusion is a common PHP security issue: en.wikipedia.org/wiki/Remote_File_Inclusion#PHP .
– Frank Farmer Feb 25, 2010 at 3:50

63
votes

strtr()

It's extremely fast, so much that you would be amazed. Internally it probably uses some crazy b-tree type structure to arrange your matches by their common prefixes. I use it with over 200 find and replace strings and it still goes through 1MB in less than 100ms. For all but trivially small strings strtr() is even significantly faster than strtolower() at doing the exact same thing, even taking character set into account. You could probably write an entire parser using successive strtr calls and it'd be faster than the usual regular expression match, figure out token type, output this or that, next regular expression kind of thing.

I was writing a text normaliser for splitting text into words, lowercasing, removing punctuation etc and strtr was my Swiss army knife, it beat the pants off regular expressions or even str_replace().

Share

edited Nov 14, 2023 at 22:25

community wiki
4 revs, 2 users 87%
thomasrutter

1 It's probably faster because it does single character replacements. – staticsan Jun 22, 2009 at 0:30

10 strtr() does not just do single-character replacements. It can replace arbitrary-length substrings with other arbitrary-length substrings, and it still seems really fast. – thomasrutter Jun 22, 2009 at 4:52

1 You mentioned it was faster than strtolower in some cases, can you prove it? I did a small benchmark and found it to be false. – The Pixel Developer Mar 6, 2010 at 21:14

1 I found that on small strings of say 80 characters it was slower than strtolower, and on large strings of say 1MB it was faster. There's probably some fixed cost type overhead each time it's called. I was simply using strtr($this->string, 'ABCDEFGHIJKLMNOPQRSTUVWXYZ', 'abcdefghijklmnopqrstuvwxyz'); The string I was operating on was some typical English text ("The quick brown fox", that sort of thing). – thomasrutter Mar 9, 2010 at 6:03

I just read this post and then did some Google searching and found: simplemachines.org/community/index.php?topic=175031.0 Any idea which one is right? – Flipper Sep 7, 2011 at 6:10

61
votes

One not so well known feature of PHP is `extract()`, a function that unpacks an associative array into the local namespace. This probably exists for the autoglobal abormination but is very useful for templating:

```
function render_template($template_name, $context, $as_s
{
    extract($context);
    if ($as_string)
        ob_start();
    include TEMPLATE_DIR . '/' . $template_name;
    if ($as_string)
        return ob_get_clean();
}
```

Now you can use `render_template('index.html', array('foo' => 'bar'))` and only `$foo` with the value `"bar"` appears in the template.

Share

edited Feb 23, 2010 at 21:18

community wiki
4 revs, 4 users 80%
Armin Ronacher

15 I was going to get angry at you for suggesting extract() in any way was a good feature. But I guess that use of it is pretty handy. I suppose it's the one time I've seen it used where it's a good idea! – thomasrutter Mar 20, 2009 at 9:59

5 extract() makes it dead simple to roll your own extremely light weight templating system. +1 – Dinah Mar 25, 2009 at 18:32

14 Its inverse, compact(), is nice as well: $a = 1; $b = 2; compact('a', 'b'); // == array('a' => $a, 'b' => $b) – user7675 Jun 6, 2009 at 21:48

2 Yes, that's probably the only really good use of extract(). – staticsan Jun 22, 2009 at 0:33

4 I would suggest not using common words as function argument names in this case, since the $context from which you extract() could contain values at the 'as_string' or 'template_name' indexes. Using EXTR_SKIP is acceptable, but only moves the problem elsewhere (i.e. if the template is expecting an

$as_string it'll get the incorrect value for it.)
– Michał Tatarynowicz Nov 3, 2009 at 18:10

52
votes

Range() isn't hidden per se, but I still see a lot of people iterating with:

```
for ($i=0; $i < $x; $i++) {
    // code...
}
```

when they could be using:

```
foreach (range(0, 12) as $number) {
    // ...
}
```

And you can do simple things like

```
foreach (range(date("Y"), date("Y")+20) as $i)
{
print "\t<option value=\"{$i}\">{$i}</option>\n";
}
```

Share

edited Aug 30, 2010 at 16:44

community wiki
3 revs, 2 users 88%
Darren Newton

3 you can do foreach ($array as $key=>$value) {} which is even simpler. – SilentGhost Jun 22, 2009 at 11:59

15 It might be a micro-optimization, but it's worth noting that for and while are much faster and less memory-intensive than foreach. – JAL Dec 19, 2009 at 6:42

3 You should try Python. It's as simple as "for i in range(12)", or you can use the more efficient xrange. – Ponkadoodle Feb 24, 2010 at 0:15

2 @SilentGhost: you'd still need an array to start with, which isn't always the case. @Newbie: elaborate please? – Alec Feb 27, 2010 at 13:19

1 @flexxy, appartently, it's NOT. Consider this: phpbench.com scroll down to `Read Loop`. – Buddy Jul 28, 2011 at 13:41

44 votes

PHP enabled webspace is usually less expensive than something with (asp).net. You *might* call that a feature ;-)

Share

edited Feb 23, 2010 at 21:19

community wiki
2 revs, 2 users 67%
VolkerK

3 It's also much cheaper to set up multiple servers if you don't have to pay for Windows Server on every one. – MiffTheFox Jun 6, 2009 at 7:29

4 Only place I know of where Windows is cost effective is at a University that gets STEEEEP discounts on the server software

in as much as it is cheaper for my dept to buy 100 copies of windows than it is to train our admins on linux (which partially makes me sad but their windows setup is clean and well setup). – dcousineau Jun 21, 2009 at 22:37

4 For now, but you have to make the switch only once, while you'll have to buy new licences sooner or later... – Bite code Oct 26, 2009 at 13:35

42 votes

The `static` keyword is useful outside of a OOP standpoint. You can quickly and easily implement 'memoization' or function caching with something as simple as:

```php
<?php
function foo($arg1)
{
    static $cache;

    if( !isset($cache[md5($arg1)]) )
    {
        // Do the work here
        $cache[md5($arg1)] = $results;
    }

    return $cache[md5($arg1)];
}
?>
```

The `static` keyword creates a variable that persists only within the scope of that function past the execution. This technique is great for functions that hit the database like `get_all_books_by_id(...)` or `get_all_categories(...)` that you would call more than once during a page load.

Caveat: Make sure you find out the best way to make a key for your hash, in just about every circumstance the `md5(...)` above is NOT a good decision (speed and output length issues), I used it for illustrative purposes. `sprintf('%u', crc32(...))` or `spl_object_hash(...)` may be much better depending on the context.

Share

edited Aug 25, 2009 at 13:53

community wiki
2 revs
dcousineau

7 Just a copied feature from C/C++ – GetFree Dec 18, 2009 at 18:12

2 @GetFree I don't think anyone would deny that almost all of PHP has been copied from C, C++, Perl, etc. – Frank Farmer Jun 26, 2010 at 21:34

1 Copying is the best thing that can be done. – NikiC Aug 30, 2010 at 16:41

1 Bad PHP. It's always copying features from others. It should write everything from scratch instead! (In case of the slightest possibility of this being taken seriously: I'm joking) – Halil Özgür Dec 7, 2011 at 9:06

42
votes

One nice feature of PHP is the **CLI**. It's not so "promoted" in the documentation but if you need routine scripts / console apps, using cron + php cli is really fast to develop!

Share

edited Feb 23, 2010 at 20:47

community wiki
2 revs, 2 users 67%
Lucacri

I should really look into this, I have several cron jobs that fetch a PHP script through `wget http://example.com...` – DisgruntledGoat Jun 28, 2010 at 19:54

CLI is also an excellent way to spot-debug, as warnings/errors will show without you having to change your error reporting preferences. – Benjamin Carlsson Sep 27, 2011 at 18:13

39 votes

Then "and print" trick

```
<?php $flag and print "Blah" ?>
```

Will echo Blah if $flag is true. DOES NOT WORK WITH ECHO.

This is very handy in template and replace the ? : that are not really easy to read.

Share

edited May 1, 2010 at 9:31

community wiki
3 revs, 3 users 84%
e-satis

24 I myself find the ternary operator much more obvious than exploiting the evaluation short-cirtcuit of a logical and. – Vicent Marti Oct 6, 2008 at 11:16

26 Actually that's the same number of characters as <?php if($flag) print "Blah" – too much php Nov 18, 2008 at 7:00

3 Parenthesis are not as easy to type as "and", especially on my bloody french keyboard ;-) – Bite code Nov 18, 2008 at 17:10

7 @all comments - This isn't about how short you can get it! It's about readability and ease of use for template people, who sometimes are not even programmers. – Tor Valamo Jan 12, 2010 at 7:47

6 I would comment that the if () statement is easier and more readable. It's certainly easier for me to get my head around than exploiting essentially a side effect of the 'and' operator in PHP, where it's easy to make a mistake (or look like it's a mistake when you read the code later). For instance, as stated this won't work the way you want with 'echo'. With if () there aren't gotchas like this. – thomasrutter Apr 10, 2010 at 2:54

37 votes

You can use **minus character in variable names** like this:

```
class style
{
  ....
  function set_bg_colour($c)
  {
    $this->{'background-color'} = $c;
  }
}
```

Why use it? No idea: maybe for a CSS model? Or some weird JSON you need to output. It's an odd feature :)

Share edited Nov 18, 2012 at 16:29

community wiki
3 revs, 3 users 90%
monk-e-boy

2 Does that work with method names? Could be useful for frameworks which use a router and I want domain.com/something-with-minuses/view/ – alex Jul 9, 2009 at 12:40

6 The curly braces allow you to access object properites that have dashes, dots, and other non-alphanumeric entities. One reason to use it is when dealing with xml, where the entity names can be dotted like in NITF/NewsML <body.content>. If you use SimpleXML, you would access it like this `$item->DataContent->body->{'body.content'}` . – Jesse Kochis Sep 20, 2009 at 19:22

2 PHP variables can take any characters when used that way, even spaces and newlines. – Newbie Feb 20, 2010 at 20:58

This would be very useful when using in SimpleXML... awesome. Thanks for sharing. – KyleFarris Sep 2, 2011 at 14:16

34 votes

HEREDOC syntax is my favourite hidden feature. Always difficult to find as you can't Google for <<< but it stops you having to escape large chunks of HTML and still allows you to drop variables into the stream.

```
echo <<<EOM
  <div id="someblock">
    <img src="{$file}" />
  </div>
EOM;
```

Share

edited Feb 23, 2010 at 20:55

community wiki
2 revs, 2 users 92%
Pablo Livardo

2 HEREDOC is my favorite way to build and use SQL statements. – bdl Jan 9, 2010 at 2:25

11 Note: the closing EOM; cannot be indented. – micahwittman Jan 9, 2010 at 6:29

Good, but pointless if you use templates. Plus it gets in the way of proper indentation. – Manos Dilaverakis Apr 23, 2010 at 9:06

7 Dear god, that is the ugliest piece of code I have ever seen. If you're using HEREDOC's, then you haven't separated presentation from logic. – Lotus Notes Jun 1, 2010 at 23:45

@Byron: You don't have to use it for presentation, it can be used for any string. See the comment from bdl. – Tom Pažourek Jul 3, 2010 at 18:47

34 votes

Probably not many know that it is possible to specify constant "variables" as default values for function parameters:

```
function myFunc($param1, $param2 = MY_CONST)
{
//code...
}
```

Strings can be used **as** if they were **arrays**:

```
$str = 'hell o World';
echo $str; //outputs: "hell o World"

$str[0] = 'H';
echo $str; //outputs: "Hell o World"

$str[4] = null;
echo $str; //outputs: "Hello World"
```

Share

edited Feb 23, 2010 at 21:21

community wiki
2 revs, 2 users 97%
jamolkhon

3 That last one is nifty. Though I have no idea when it would be better than some other method of removing a character. +1 nontheless – George Mauer Mar 20, 2009 at 15:32

3 It's probably more efficient than calling a function to do it. Strings are normally stored contiguously in memory, so getting to $str[4] is trivial. Storing strings as arrays of characters is common to most languages that derive from C. – sjobe Jun 22, 2009 at 21:02

1 You don't have to use a defined constant as a default value. The following is also perfectly valid: function foot($param1,

$default = array('key'=>'value'), $default_s = 'String', $default_i = 10, $default_b = false). However you are correct in noting you can't use a variable as a default argument. – dcousineau Jun 23, 2009 at 21:31

@dcousineau Your example is perfectly valid since array() is not a function but a language construct. Function calls are not allowed as default values for arguments. – Jamol Jul 20, 2009 at 9:48

4 Be careful with treating strings as arrays if you have multi-byte strings (foreign languages, etc.) – philfreo Apr 12, 2011 at 5:48

---

33 votes

The single most useful thing about PHP code is that if I don't quite understand a function I see I can look it up by using a browser and typing:

> http://php.net/function

Last month I saw the "range" function in some code. It's one of the hundreds of functions I'd managed to never use but turn out to be really useful:

> http://php.net/range

That url is an alias for https://www.php.net/manual/en/function.range.php. That simple idea, of **mapping functions and keywords to urls**, is awesome.

I wish other languages, frameworks, databases, operating systems has as simple a mechanism for looking up documentation.

Share

edited Nov 17, 2023 at 20:38

community wiki
4 revs, 2 users 90%
artlung

5 `range()` can be useful for `foreach( range(1, 10) as $i) { };` – alex Jul 9, 2009 at 12:41

If you have FireFox; just type `PHP function` in the address bar it will do a Google 'I'm feeling lucky' search and you almost always end up on the right php documentation page. – Kolky Mar 18, 2010 at 11:03

30 votes

Fast block comments

```
/*
    die('You shall not pass!');
//*/


//*
    die('You shall not pass!');
//*/
```

These comments allow you to toggle if a code block is commented with one character.

Share

edited Sep 15, 2010 at 10:35

community wiki
14 revs, 3 users 79%
Sam152

14 This isn't really specific to PHP. This works in any language that supports `// ...` line comments and `/* ... */` block comments. – Jordan Ryan Moore Dec 7, 2009 at 15:44

any code cleanup utilities end up hating you for using this... ;) – Talvi Watia Jan 29, 2010 at 10:41

3 I've also used `/** /` before and `/**/` after. You can toggle the block by removing and adding the space in the first. This has an added benefit of working with CSS (and other languages that do not support `// ...` comments). – kingjeffrey Jun 9, 2010 at 22:35

FWIW, link to the original article aleembawany.com/2009/01/27/lazy-block-comment-trick – aleemb Sep 14, 2010 at 11:42

@aleemb, refrain from making any further edits to this question. – Sam Becker Sep 15, 2010 at 10:30

29
votes

My list.. most of them fall more under the "hidden features" than the "favorite features" (I hope!), and not all are useful, but .. yeah.

```
// swap values. any number of vars works, obviously
list($a, $b) = array($b, $a);

// nested list() calls "fill" variables from multidim ar
```

```
$arr = array(
  array('aaaa', 'bbb'),
  array('cc', 'd')
);
list(list($a, $b), list($c, $d)) = $arr;
echo "$a $b $c $d"; // -> aaaa bbb cc d

// list() values to arrays
while (list($arr1[], $arr2[], $arr3[]) = mysql_fetch_row
// or get columns from a matrix
foreach($data as $row) list($col_1[], $col_2[], $col_3[]

// abusing the ternary operator to set other variables a
$foo = $condition ? 'Yes' . (($bar = 'right') && false)
'left') && false);
// boolean False cast to string for concatenation become
// you can also use list() but that's so boring ;-)
list($foo, $bar) = $condition ? array('Yes', 'right') :
```

You can nest ternary operators too, comes in handy sometimes.

```
// the strings' "Complex syntax" allows for *weird* stuf
// given $i = 3, if $custom is true, set $foo to $P['siz
$C['size3']:
$foo = ${$custom?'P':'C'}['size'.$i];
$foo = $custom?$P['size'.$i]:$C['size'.$i]; // does the
;-)
// similarly, splitting an array $all_rows into two arra
based
// on some field 'active' in the sub-arrays:
foreach ($all_rows as $row) ${'data'.($row['active']?1:0

// slight adaption from another answer here, I had to tr
could
// abuse as variable names.. turns out, way too much...
$string = 'f.> <!-? o+';
${$string} = 'asdfasf';
echo ${$string}; // -> 'asdfasf'
echo $GLOBALS['f.> <!-? o+']; // -> 'asdfasf'
// (don't do this. srsly.)
```

```
${''} = 456;
echo ${''}; // -> 456
echo $GLOBALS['']; // -> 456
// I have no idea.
```

Right, I'll stop for now :-)

Hmm, it's been a while..

```
// just discovered you can comment the hell out of php:
$q/* snarf */=/* quux */$_GET/* foo */[/* bar */'q'/* ba
```

So, just discovered you can pass any string as a method name IF you enclose it with curly brackets. You can't define any string as a method alas, but you can catch them with __call(), and process them further as needed. Hmmm....

```
class foo {
  function __call($func, $args) {
    eval ($func);
  }
}

$x = new foo;
$x->{'foreach(range(1, 10) as $i) {echo $i."\n";}'}();
```

Found this little gem in Reddit comments:

```
$foo = 'abcde';
$strlen = 'strlen';
echo "$foo is {$strlen($foo)} characters long."; // "abc
long."
```

You can't call functions inside {} directly like this, but you *can* use variables-holding-the-function-name and call those! (*and* you can use variable variables on it, too)

Share

edited Nov 21, 2010 at 16:57

community wiki
7 revs, 2 users 75%
MSpreij

2 Please don't overuse the ternary comparison operator; this leads to code obfuscation. – staticsan Jun 22, 2009 at 0:24

Wow. That could totally hose your $GLOBALS list. Bad practice. Seriously. – Talvi Watia Jan 29, 2010 at 11:22

Is there an obfuscated PHP contest yet? – Lotus Notes Jun 1, 2010 at 23:54

Well, trick with swap - impressive and useful, thanks. – user680786 Apr 28, 2011 at 20:45

1 `${''} = 456;` hahaha.... quite the abuse. – Skurmedel Jun 28, 2011 at 14:42

26 votes

Array manipulation.
Tons of tools for working with and manipulating arrays. It may not be unique to PHP, but I've never worked with a language that made it so easy.

Share

answered Sep 14, 2008 at 17:51

community wiki
MattBelanger

like what for example? It seems to me like the functions are all awkwardly named and positioned in the global namespace. Plus I can't think of anything thats not just as easy in another language except for maybe $arr[] = $newvalue for adding values - thats cool – George Mauer Sep 14, 2008 at 18:29

8 Well, the PHP array is a datastructure that can be used easily as a stack, queue, deque, list, hashtable, etc. It's pretty flexible indeed for most common needs, without resorting to anything else but array_* functions. – Camilo Díaz Repka Sep 19, 2008 at 14:22

6 Python does arrays (as lists and tuples) much better than PHP does. – too much php Nov 18, 2008 at 7:08

26 votes

I'm a bit like you, I've coded PHP for over 8 years. I had to take a .NET/C# course about a year ago and I really enjoyed the C# language (hated ASP.NET) but it made me a better PHP developer.

PHP as a language is pretty poor, but, I'm extremely quick with it and the LAMP stack is awesome. The end product far outweighs the sum of the parts.

That said, in answer to your question:

http://uk.php.net/SPL

I love the **SPL**, the collection class in C# was something that I liked as soon as I started with it. Now I can have my cake and eat it.

Andrew

Share

edited Feb 23, 2010 at 20:51

community wiki
2 revs, 2 users 95%
Andrew Taylor

24
votes

I'm a little surprised no-one has mentioned it yet, but one of my favourite tricks with arrays is using the plus operator. It is a little bit like `array_merge()` but a little simpler. I've found it's usually what I want. In effect, it takes all the entries in the RHS and makes them appear in a copy of the LHS, overwriting as necessary (i.e. it's non-commutative). Very useful for starting with a "default" array and adding some real values all in one hit, whilst leaving default values in place for values not provided.

Code sample requested:

```
// Set the normal defaults.
$control_defaults = array( 'type' => 'text', 'size' => 3

// ... many lines later ...

$control_5 = $control_defaults + array( 'name' => 'surna
// This is the same as:
// $control_5 = array( 'type' => 'text', 'name' => 'surn
```

Share

edited Apr 27, 2010 at 1:22

community wiki
4 revs, 2 users 94%
staticsan

`$defaults` should be `$control_defaults` – xkeshav Apr 23, 2010 at 8:33

3 I think it's not as clear as array_merge when you have a lot of code to maintain. At least, when you use, the array_merge function, it's evident that you're dealing with arrays. – Sylvain May 6, 2010 at 18:19

And that fact that you're doing `... + array( ...` isn't enough to point this out? :-) – staticsan May 7, 2010 at 6:24

What version of PHP do you need for this? – Lotus Notes Jun 1, 2010 at 23:50

This is a great feature and it should be noted that the array on the "right" side of the "+" will not over-write existing keys of the array to the "left" side of the "+". – Wil Moore III Sep 1, 2010 at 21:36

21 votes

Here's one, I like how **setting default values on function parameters** that aren't supplied is much easier:

```
function MyMethod($VarICareAbout, $VarIDontCareAbout = '
```

Share

edited Feb 23, 2010 at 21:09

community wiki
3 revs, 2 users 83%
George Mauer

4 Funnily enough I saw this "hidden feature" in Google Reader last week. I don't get what's hidden about it - it's basic syntax. Try if($var = true) for example. – Ross Oct 13, 2008 at 15:28

8 Easier than what? Most language have this feature. – Christian Davén Apr 8, 2009 at 14:28

10 Easier than C# (and I think C++, and Java) – George Mauer Apr 8, 2009 at 15:40

8 C++ does support default parameter values. – sjobe Jun 22, 2009 at 20:45

2 c# doesn't support default values at all. you have to write an overloaded function and always declare the value which is just plain cumbersome – DeveloperChris Jan 9, 2010 at 6:02

21 votes

Quick and dirty is the default.
The language is filled with useful shortcuts, This makes PHP the perfect candidate for (small) projects that have a short time-to-market. Not that clean PHP code is impossible, it just takes some extra effort and experience.

But I love PHP because it lets me express what I want without typing an essay.

PHP:

```
if (preg_match("/cat/","one cat")) {
   // do something
}
```

JAVA:

```
import java.util.regex.*;
Pattern p = Pattern.compile("cat");
Matcher m = p.matcher("one cat")
if (m.find()) {
  // do something
}
```

And yes, that includes not typing **Int**.

Share

edited Sep 13, 2011 at 17:52

community wiki
4 revs, 3 users 93%
Bob Fanger

4 you should use strpos instead: if (false !== strpos("one cat", "cat")) { – OIS Apr 12, 2009 at 14:06

17 @OIS the purpose of his example was to illustrate and compare the running of a quick regex match, not how to find the string "cat" in "one cat". – dcousineau Jun 23, 2009 at 21:24

19 Java: if (Pattern.matches("cat", "one cat")) { // do something } Stop complaining about java if you don't know it.
– whiskeysierra Dec 21, 2009 at 13:52

3 +1 Willi how do you do preg_replace('/([<[]!--\s*)(\S*?)(\s*--[>]]?)/se', "\$this->Choose('\\1','\\2','\\3',\$data)", $text); in Java? This finds a comment in the input text and then calls a function with the matched elements which in this case $this-

>choose(...) decides what to replace the match with and returns the results. – DeveloperChris Jan 9, 2010 at 6:49

1 Regex is PHP is pretty crappy... I'd much rather have Perl or JavaScript style Regex where the `//` is built into the language. – cdmckay Apr 19, 2010 at 20:26