PHP 7.3.0.beta3 Released

Keyboard Shortcuts

?
: This help

j
: Next menu item

k
: Previous menu item

g p
: Previous man page

g n
: Next man page

G
: Scroll to bottom

g g
: Scroll to top

echo

(PHP 4, PHP 5, PHP 7)

echo — Output one or more strings

Description ¶

void **echo** (string `$arg1` [, string `$...`])

Outputs all parameters. No additional newline is appended.

echo is not actually a function (it is a language construct), so you are not required to use parentheses with it. *echo* (unlike some other language constructs) does not behave like a function, so it cannot always be used in the context of a function. Additionally, if you want to pass more than one parameter to *echo*, the parameters must not be enclosed within parentheses.

echo also has a shortcut syntax, where you can immediately follow the opening tag with an equals sign. Prior to PHP 5.4.0, this short syntax only works with the short_open_tag configuration setting enabled.

```
I have <?=$foo?> foo.
```

The major differences to *print* are that *echo* accepts an argument list and doesn't have a return value.

Parameters ¶

`arg1`

The parameter to output.

`...`

Return Values ¶

No value is returned.

Examples ¶

Example #1 *echo* examples

```php
<?php
echo "Hello World";

echo "This spans
multiple lines. The newlines will be
output as well";

echo "This spans\nmultiple lines. The newlines will be\noutput as well.";

echo "Escaping characters is done \"Like this\".";

// You can use variables inside of an echo statement
$foo = "foobar";
$bar = "barbaz";

echo "foo is $foo"; // foo is foobar

// You can also use arrays
$baz = array("value" => "foo");

echo "this is {$baz['value']} !"; // this is foo !

// Using single quotes will print the variable name, not the value
echo 'foo is $foo'; // foo is $foo

// If you are not using any other characters, you can just echo variables
echo $foo;          // foobar
echo $foo,$bar;     // foobarbarbaz

// Strings can either be passed individually as multiple arguments or
// concatenated together and passed as a single argument
echo 'This ', 'string ', 'was ', 'made ', 'with multiple parameters.', chr(10);
echo 'This ' . 'string ' . 'was ' . 'made ' . 'with concatenation.' . "\n";

echo <<<END
This uses the "here document" syntax to output
multiple lines with $variable interpolation. Note
that the here document terminator must appear on a
line with just a semicolon. no extra whitespace!
END;

// Because echo does not behave like a function, the following code is invalid.
($some_var) ? echo 'true' : echo 'false';

// However, the following examples will work:
```

```
($some_var) ? print 'true' : print 'false'; // print is also a construct, but
                                            // it behaves like a function, so
                                            // it may be used in this context.
echo $some_var ? 'true': 'false'; // changing the statement around
?>
```

Notes ¶

> **Note**: Because this is a language construct and not a function, it cannot be called using variable functions.

Tip

A benefit to passing in multiple arguments over using concatenation in **echo** regards the precedence of the period operator in PHP. If multiple arguments are passed in, then parentheses will not be required to enforce precedence:

```
<?php
echo "Sum: ", 1 + 2;
echo "Hello ", isset($name) ? $name : "John Doe", "!";
```

With concatenation, the period operator has a higher precedence than both the addition and ternary operators, and so parentheses must be used for the correct behaviour:

```
<?php
echo 'Sum: ' . (1 + 2);
echo 'Hello ' . (isset($name) ? $name : 'John Doe') . '!';
```

See Also ¶

- print - Output a string
- printf() - Output a formatted string
- flush() - Flush system output buffer
- Heredoc syntax

add a note

User Contributed Notes 3 notes

up
down
17
pemapmodder1970 at gmail dot com ¶
1 year ago

Passing multiple parameters to echo using commas (',')is not exactly identical to using the concatenation operator ('.'). There are two notable differences.

First, concatenation operators have much higher precedence. Referring to http://php.net/operators.precedence, there are many operators with lower precedence than concatenation, so it is a good idea to use the multi-argument form instead of passing concatenated strings.

```
<?php
echo "The sum is " . 1 | 2; // output: "2". Parentheses needed.
```

```
echo "The sum is ", 1 | 2; // output: "The sum is 3". Fine.
?>
```

Second, a slightly confusing phenomenon is that unlike passing arguments to functions, the values are evaluated one by one.

```
<?php
function f($arg){
  var_dump($arg);
  return $arg;
}
echo "Foo" . f("bar") . "Foo";
echo "\n\n";
echo "Foo", f("bar"), "Foo";
?>
```

The output would be:

```
string(3) "bar"FoobarFoo

Foostring(3) "bar"
barFoo
```

It would become a confusing bug for a script that uses blocking functions like sleep() as parameters:

```
<?php
while(true){
  echo "Loop start!\n", sleep(1);
}
?>
```

vs

```
<?php
while(true){
  echo "Loop started!\n" . sleep(1);
}
?>
```

With ',' the cursor stops at the beginning every newline, while with '.' the cursor stops after the 0 in the beginning every line (because sleep() returns 0).

up
down
-8

Jamie Robinson ¶

2 years ago

The {} syntax is useful for printing non array variables as well, an example to illustrate:

```
<?php
$foo = "foobar";
$bar = "barbaz";

//Will produce the error: Undefined variable: $foo_
echo "$foo_$bar";
```

```
//Will print the intended string: "foobar_barbaz"
echo "{$foo}_$bar";
?>
```

Could even be worth getting into the habit of enclosing all variables in {} when writing echo strings, to be on the safe side.

up
down
-61
214363570 at qq dot com ¶
1 year ago

```
Dear:
      Is there a offical function like echoln(), such as

     function echoln($s=""){
            echo $s."\n";
     }

     $str = "i love php";
     echoln($str);
     echoln($str);
     echoln($str);
     echoln($str);

     not is:
     echo $str."\n";
     echo $str."\n";
     echo $str."\n";
     echo $str."\n";
     echo $str."\n";

     Thank you.

     Best regards.
```

add a note

- String Functions
  - addcslashes
  - addslashes
  - bin2hex
  - chop
  - chr
  - chunk_split
  - convert_cyr_string
  - convert_uudecode
  - convert_uuencode
  - count_chars
  - crc32
  - crypt
  - echo
  - explode
  - fprintf
  - get_html_translation_table
  - hebrev

- hebrevc
- hex2bin
- html_entity_decode
- htmlentities
- htmlspecialchars_decode
- htmlspecialchars
- implode
- join
- lcfirst
- levenshtein
- localeconv
- ltrim
- md5_file
- md5
- metaphone
- money_format
- nl_langinfo
- nl2br
- number_format
- ord
- parse_str
- print
- printf
- quoted_printable_decode
- quoted_printable_encode
- quotemeta
- rtrim
- setlocale
- sha1_file
- sha1
- similar_text
- soundex
- sprintf
- sscanf
- str_getcsv
- str_ireplace
- str_pad
- str_repeat
- str_replace
- str_rot13
- str_shuffle
- str_split
- str_word_count
- strcasecmp
- strchr
- strcmp
- strcoll
- strcspn
- strip_tags
- stripcslashes
- strpos
- stripslashes
- stristr
- strlen

- strnatcasecmp
- strnatcmp
- strncasecmp
- strncmp
- strpbrk
- strpos
- strrchr
- strrev
- strripos
- strrpos
- strspn
- strstr
- strtok
- strtolower
- strtoupper
- strtr
- substr_compare
- substr_count
- substr_replace
- substr
- trim
- ucfirst
- ucwords
- vfprintf
- vprintf
- vsprintf
- wordwrap

- My PHP.net
- Contact
- Other PHP.net sites
- Mirror sites
- Privacy policy