globals

$GLOBALS

References all variables available in global scope

ENTITY

An associative `array` containing references to all variables which are currently defined in the global scope of the script. The variable names are the keys of the array.

ENTITY

`$GLOBALS` example

```
<?php

function test()
{
    $foo = "local variable";

    echo '$foo in global scope: ' . $GLOBALS["foo"] . "\n";
    echo '$foo in current scope: ' . $foo . "\n";
}

$foo = "Example content";
test();

?>
```

ENTITY

```
$foo in global scope: Example content
$foo in current scope: local variable
```

As of PHP 8.1.0, write access to the entire `$GLOBALS` array is no longer supported:

writing entire `$GLOBALS` will result in error.

```
<?php

// Generates compile-time error:
$GLOBALS = [];
$GLOBALS += [];
$GLOBALS =& $x;
$x =& $GLOBALS;
unset($GLOBALS);
array_pop($GLOBALS);
// ...and any other write/read-write operation on $GLOBALS

?>
```

ENTITY ENTITY

Unlike all of the other superglobals, `$GLOBALS` has essentially always been available in PHP.

As of PHP 8.1.0, `$GLOBALS` is now a read-only copy of the global symbol table. That is, global variables cannot be modified via its copy. Previously, `$GLOBALS` array is excluded from the usual by-value behavior of PHP arrays and global variables can be modified via its copy.

```
<?php

// Before PHP 8.1.0
$a = 1;

$globals = $GLOBALS; // Ostensibly by-value copy
$globals['a'] = 2;
var_dump($a); // int(2)
```

```
// As of PHP 8.1.0
// this no longer modifies $a. The previous behavior violated by-value semantics.
$globals = $GLOBALS;
$globals['a'] = 1;

// To restore the previous behavior, iterate its copy and assign each property back to
$GLOBALS.
foreach ($globals as $key => $value) {
    $GLOBALS[$key] = $value;
}

?>
```

□□□ #2
□ □□□ □□□ 2026-10-31 17:46:35 UTC
□ □□□ □□□ 2026-10-31 17:46:36 UTC