Changes to variable handling

- Changes to the handling of indirect variables, properties, and methods
- Changes to list handling
  - list no longer assigns variables in reverse order
  - Empty list assignments have been removed
  - list cannot unpack strings
- Array ordering when elements are automatically created during by reference assignments has changed
- Parentheses around function arguments no longer affect behaviour

PHP 7 now uses an abstract syntax tree when parsing source files. This has permitted many improvements to the language which were previously impossible due to limitations in the parser used in earlier versions of PHP, but has resulted in the removal of a few special cases for consistency reasons, which has resulted in backward compatibility breaks. These cases are detailed in this section.

Changes to the handling of indirect variables, properties, and methods

Indirect access to variables, properties, and methods will now be evaluated strictly in left-to-right order, as opposed to the previous mix of special cases. The table below shows how the order of evaluation has changed.

Expression	PHP 5 interpretation	PHP 7 interpretation
`$$foo['bar']['baz']`	`${$foo['bar']['baz']}`	`($$foo)['bar']['baz']`
`$foo-ENTITY$bar['baz']`	`$foo-ENTITY{$bar['baz']}`	`($foo-ENTITY$bar)['baz']`
`$foo-ENTITY$bar['baz']()`	`$foo-ENTITY{$bar['baz']}()`	`($foo-ENTITY$bar)['baz']()`
`Foo::$bar['baz']()`	`Foo::{$bar['baz']}()`	`(Foo::$bar)['baz']()`

Old and new evaluation of indirect expressions

Code that used the old right-to-left evaluation order must be rewritten to explicitly use that evaluation order with curly braces (see the above middle column). This will make the code both forwards compatible with PHP 7.x and backwards compatible with PHP 5.x.

This also affects the ENTITY keyword. The curly brace syntax can be used to emulate the previous behaviour if required:

```
<?php
function f() {
    // Valid in PHP 5 only.
    global $$foo->bar;

    // Valid in PHP 5 and 7.
    global ${$foo->bar};
}
?>
```

Changes to `list` handling

`list` no longer assigns variables in reverse order

`list` will now assign values to variables in the order they are defined, rather than reverse order. In general, this only affects the case where `list` is being used in conjunction with the array `[]` operator, as shown below:

```
<?php
list($a[], $a[], $a[]) = [1, 2, 3];
var_dump($a);
?>
```

ENTITY

```
array(3) {
  [0]=>
```

```
  int(3)
  [1]=>
  int(2)
  [2]=>
  int(1)
}
```

ENTITY

```
array(3) {
  [0]=>
  int(1)
  [1]=>
  int(2)
  [2]=>
  int(3)
}
```

In general, it is recommended not to rely on the order in which `list` assignments occur, as this is an implementation detail that may change again in the future.

Empty `list` assignments have been removed

`list` constructs can no longer be empty. The following are no longer allowed:

```
<?php
list() = $a;
list(,,) = $a;
list($x, list(), $y) = $a;
?>
```

`list` cannot unpack `string`s

`list` can no longer unpack `string` variables. `str_split` should be used instead.

Array ordering when elements are automatically created during by reference

assignments has changed

The order of the elements in an array has changed when those elements have been automatically created by referencing them in a by reference assignment. For example:

```
<?php
$array = [];
$array["a"] =& $array["b"];
$array["b"] = 1;
var_dump($array);
?>
```

ENTITY

```
array(2) {
  ["b"]=>
  &int(1)
  ["a"]=>
  &int(1)
}
```

ENTITY

```
array(2) {
  ["a"]=>
  &int(1)
  ["b"]=>
  &int(1)
}
```

Parentheses around function arguments no longer affect behaviour

In PHP 5, using redundant parentheses around a function argument could quiet strict standards warnings when the function argument was passed by reference. The warning will now always be issued.

```
<?php
function getArray() {
    return [1, 2, 3];
}

function squareArray(array &$a) {
    foreach ($a as &$v) {
        $v **= 2;
    }
}

// Generates a warning in PHP 7.
squareArray((getArray()));
?>
```

ENTITY

```
Notice: Only variables should be passed by reference in /tmp/test.php on line 13
```

□□□ #2
□ □□□ □□□ 2026-10-31 17:33:16 UTC
□ □□□ □□□ 2026-10-31 17:33:17 UTC