PHP - Spread Operator

PHP recognizes the three dots symbol (...) as the **spread operator**. The spread operator is also sometimes called the **splat operator**. This operator was first introduced in PHP version 7.4. It can be effectively used in many cases such as unpacking arrays.

Example 1

In the example below, the elements in $arr1 are inserted in $arr2 after a list of its own elements.

</> Open Compiler

```
<?php
   $arr1 = [4,5];
   $arr2 = [1,2,3, ...$arr1];

   print_r($arr2);
?>
```

It will produce the following **output** –

```
Array
(
  [0] => 1
  [1] => 2
  [2] => 3
  [3] => 4
  [4] => 5
)
```

Example 2

The Spread operator can be used more than once in an expression. For example, in the following code, a third array is created by expanding the elements of two arrays.

</> Open Com

```php
<?php
   $arr1 = [1,2,3];
   $arr2 = [4,5,6];
   $arr3 = [...$arr1, ...$arr2];

   print_r($arr3);
?>
```

It will produce the following **output** –

```
Array
(
  [0] => 1
  [1] => 2
  [2] => 3
  [3] => 4
  [4] => 5
  [5] => 6
)
```


Example 3

Note that the same result can be obtained with the use of **array_merge()** function, as shown below –

```php
</>  Open Compiler

<?php
   $arr1 = [1,2,3];
   $arr2 = [4,5,6];
   $arr3 = array_merge($arr1, $arr2);

   print_r($arr3);
?>
```

It will produce the same **output** –

```
Array
(
  [0] => 1
  [1] => 2
  [2] => 3
  [3] => 4
  [4] => 5
  [5] => 6
)
```

However, the use of (...) operator is much more efficient as it avoids the overhead a function call.

Example 4

PHP 8.1.0 also introduced another feature that allows using named arguments after unpacking the arguments. Instead of providing a value to each of the arguments individually, the values from an array will be unpacked into the corresponding arguments, using ... (three dots) before the array.

</> Open Compiler

```php
<?php
   function  myfunction($x, $y, $z=30) {
      echo "x = $x  y = $y  z = $z";
   }

   myfunction(...[10, 20], z:30);
?>
```

It will produce the following **output** –

```
x = 10  y = 20  z = 30
```

Example 5

In the following example, the return value of a function is an array. The array elements are then spread and unpacked.

</> Open Compiler

```php
<?php
   function get_squares() {
      for ($i = 0; $i < 5; $i++) {
         $arr[] = $i**2;
      }
      return $arr;
   }
   $squares = [...get_squares()];
   print_r($squares);
?>
```

It will produce the following **output** –

```
Array
(
  [0] => 0
  [1] => 1
  [2] => 4
  [3] => 9
  [4] => 16
)
```