PHP - Named Arguments

The feature of Named Arguments has been introduced in PHP with the version 8.0. It is an extension of the existing mechanism of passing positional arguments to a function while calling.

By default, values of passed arguments are copied to the corresponding formal arguments at the same position. This feature of named arguments in PHP makes it possible to pass the value based on the parameter name instead of the position.

If we have a function defined as follows –

```
function myfunction($x, $y) {
   statement1;
   statement2;
   . . .
}
```

and it is called as –

```
myfunction(10, 20);
```

In this case, the values are passed to the variables "x" and "y" in the order of declaration. It means, the first value to the first argument, second value to second argument and so on. The variables "x" and "y" are positional arguments.

To pass the values by named arguments, specify the parameter name to which argument the value is to be passed. The name of the parameter is the name of formal argument without the "$" symbol. The value to be passed is put in front of the ":" symbol.

```
myfunction(x:10, y:20);
```

Example

Here is the code that demonstrates how you can use **named arguments** in PHP –

</> Open Compiler

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

```
   }

   myfunction(x:10, y:20);
?>
```

It will produce the following **output** –

```
x = 10  y = 20
```

Using **named arguments** makes it possible to pass the values in any order, and not necessarily in the same order in which the arguments are declared in the function definition. We can call **myfunction()** as shown below and it will produce the same result.

```
myfunction(y:20, x:10);
```

With this feature, the arguments become order-independent and self-documenting. It also makes it possible to skip the arguments with default values arbitrarily.

Combining Named Arguments with Positional Arguments

Named arguments can be combined with positional arguments, with the condition that, the named arguments must come after the positional arguments.

Example

</> Open Compiler

```
<?php
   function  myfunction($x, $y, $z) {
      echo "x = $x  y = $y  z = $z";
   }
   myfunction(10, z:20, y:30);
?>
```

It will produce the following **output** –

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

However, if you try to treat $z as a positional argument,

```
myfunction(x:10, y:20, 30);
```

In this case, PHP will encounter the following **error** −

```
PHP Fatal error:  Cannot use positional argument after
named argument in hello.php on line 7
```


Passing Named Arguments from an Array

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

Example

</> Open Compiler

```
<?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
```

Note that passing the same parameter multiple times results in an exception as follows −

```
myfunction(x:10, z:20, x:20);
```

Error −

```
PHP Fatal error:  Uncaught Error: Named parameter $x
overwrites previous argument in hello.php:7
```