PHP - Arrow Functions

Arrow functions were introduced in PHP 7.4 version. Arrow functions provide a simpler and more concise syntax for writing anonymous functions. With PHP 7.4, a keyword "**fn**" has been introduced for defining arrow functions, instead of the conventional use of the "**function**" keyword.

```
fn (argument_list) => expr
```

- There is only one expression after the "**=>**" symbol, and its value is the return value of the arrow function.
- The arrow function doesn't have an explicit **return** statement.
- Like in the anonymous function, the arrow function is assigned to a variable for it to be called.

Example

The following example demonstrates how you can use the arrow function in PHP –

</> Open Compiler

```
<?php
   $add = fn ($a, $b) => $a + $b;

   $x = 10;
   $y = 20;
   echo " x: $x y: $y Addition: " . $add($x, $y);
?>
```

It will produce the following **output** –

```
x: 10 y: 20 Addition: 30
```

Using the Arrow Function as a Callback Function

We can also use the arrow function as a callback function. Callback functions are used as one of the arguments of another function. The arrow function is executed on the fly and the value of the expression after "=>" becomes the argument of the parent function, which may be either a built-in or a user-defined function.

Example

In this example, we use an arrow function inside usort() function, a built_in function that sorts an array by values using a user-defined comparison function.

</> Open Compiler

```
<?php
   $arr = [10,3,70,21,54];
   usort ($arr, fn ($x , $y) => $x > $y);

   foreach ($arr as $x){
      echo $x . "\n";
   }
?>
```

It will produce the following **output** –

```
3
10
21
54
70
```


Accessing Variables from the Parent Scope

Arrow functions can automatically access variables from the parent scope. Unlike the anonymous functions, the "**use**" keyword is not necessary for it to act as a closure. When a variable used in the expression is defined in the parent scope, it will be implicitly captured by-value.

</> Open Compiler

```php
<?php
   $maxmarks=300;
   $percent=fn ($marks) => $marks*100/$maxmarks;

   $m = 250;
   echo "Marks = $m Percentage = ". $percent($m);
?>
```

It will produce the following **output** –

```
Marks = 250 Percentage = 83.333333333333
```

Example

Arrow functions capture variables by value automatically, even when nested.

In the following example, an arrow function is defined in the expression part of another arrow function.

</> Open Compiler

```php
<?php
   $z = 1;
   $fn = fn($x) => fn($y) => $x * $y + $z;
   $x = 5;
   $y = 10;
   echo "x:$x y:$y \n";
   echo "Result of nested arrow functions: " . ($fn($x)($y));
?>
```

It will produce the following **output** –

```
x:5 y:10
Result of nested arrow functions: 51
```

Just like anonymous functions, the arrow function syntax allows arbitrary function signatures, including parameter and return types, default values, variadics, as well as by-reference passing and returning.