PHP - Anonymous Functions

What are Anonymous Functions?

PHP allows defining anonymous functions. Normally, when we define a function in PHP, we usually provide it a name which is used to call the function whenever required. In contrast, an **anonymous function** is a function that doesn't have any name specified at the time of definition. Such a function is also called **closure** or **lambda function**.

Sometimes, you may want a function for one time use only. The most common use of anonymous functions is to create an **inline callback function**.

Anonymous functions are implemented using the **Closure** class. Closure is an anonymous function that closes over the environment in which it is defined.

The **syntax** for defining an anonymous function is as follows –

```
$var=function ($arg1, $arg2) { return $val; };
```

Note that there is no function name between the **function** keyword and the opening parenthesis, and the fact that there is a semicolon after the function definition. This implies that anonymous function definitions are expressions. When assigned to a variable, the anonymous function can be called later using the variable's name.

Example

Take a look at the following example –

</> Open Compiler

```
<?php
   $add = function ($a, $b) {
      return "a:$a b:$b addition: " . $a+$b;
   };
   echo $add(5,10);
?>
```

It will produce the following **output** –

```
a:5 b:10 addition: 15
```

Anonymous Function as a Callback

Anonymous functions are often used as callbacks. Callback functions are used as one of the arguments of another function. An anonymous function is executed on the fly and its return value 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 anonymous function inside the **usort()** function, a built in function that sorts an array by values using a user-defined comparison function.

</> Open Compiler

```php
<?php
   $arr = [10,3,70,21,54];
   usort ($arr, function ($x , $y) {
      return $x > $y;
   });
   foreach ($arr as $x){
      echo $x . "\n";
   }
?>
```

It will produce the following **output** –

```
3
10
21
54
70
```

Example

The following example uses an anonymous function to calculate the cumulative sum after successive numbers in an array. Here, we use the **array_walk()** function. This function applies a user defined function to each element in the array.

Open Compiler

```
<?php
   $arr=array(1,2,3,4,5);
   array_walk($arr, function($n){
      $s=0;
      for($i=1;$i<=$n;$i++){
         $s+=$i;
      }
      echo "Number: $n Sum: $s". PHP_EOL;
   });
?>
```

It will produce the following **output** –

```
Number: 1 Sum: 1
Number: 2 Sum: 3
Number: 3 Sum: 6
Number: 4 Sum: 10
Number: 5 Sum: 15
```


Anonymous Function as Closure

Closure is also an anonymous function that can access the variables outside its scope with the help of the "**use**" keyword.

Example

Take a look a the following example –

Open Compiler

```
<?php
   $maxmarks=300;
   $percent=function ($marks) use ($maxmarks) {
      return $marks*100/$maxmarks;
   };
```

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

It will produce the following **output** –

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