PHP - Passing Functions

To a function in PHP, in addition to scalar types, arrays, and objects, you can also pass a function as one of its arguments. If a function is defined to accept another function as an argument, the passed function will be invoked inside it. PHP's standard library has certain built-in functions of this type, where one of the arguments to be passed is a function, which may be another built-in function or even a user defined function.

array_map

The array_map() is one of the built-in functions. The first argument to this function is a callback function. There may be one or more arrays as the other arguments. The callback function is applied to all the elements of arrays.

```
array_map(?callable $callback, array $array, array ...$arrays): array
```

The array_map() function returns an array. It contains the result of applying the callback function to the corresponding elements of arrays passed as other arguments.

Example

In the following example, we have a square() function that computes the square of a number passed to it. This function in turn is used as an argument for array_map() function, along with another array of numbers. Each number is successively passed to the squares() function. The resultant array is a list of squares.

</> Open Compiler

```
<?php
   function square($number) {
      return $number * $number;
   }

   $arr = [1, 2, 3, 4, 5];
   $squares = array_map('square', $arr);
   var_dump($squares);
?>
```

It will produce the following **output** –

```
array(5) {
  [0]=>
  int(1)
  [1]=>
  int(4)
  [2]=>
  int(9)
  [3]=>
  int(16)
  [4]=>
  int(25)
}
```

call_user_func

Another example of passing a function to another function is call_user_func(). As the name suggests, it calls another user defined callback function, and the other arguments are passed to the callback.

```
call_user_func(callable $callback, mixed ...$args): mixed
```

Example

In the example below, the square() function is invoked repeatedly, passing each number in an array.

</> Open Compiler

```
<?php
   function square($number) {
      return $number * $number;
   }
   $arr = [1, 2, 3, 4, 5];
   foreach($arr as $a) {
      echo "square of $a:" . call_user_func("square", $a). PHP_EOL;
   }
?>
```

It will produce the following **output** –

```
square of 1:1
square of 2:4
square of 3:9
square of 4:16
square of 5:25
```


usort

As another example of passing function, we take a look a usort() function.

```
usort(array &$array, callable $callback): true
```

The first parameter is an array. The array is sorted as per the callback function, which is the second parameter.

The callback parameter is a comparison function must return an integer less than, equal to, or greater than zero if the first argument is considered to be respectively less than, equal to, or greater than the second.

Example

Here is an example. First we have a **mysort()** function. It compares two numbers and returns "-1", "0" or "1" if the first number is less than, equal to or greater than second number.

The first argument to **usort()** is the mysort() function, and the second one is an array. To begin with, the first two numbers are passed to mysort(). If it returns 1, they are swapped. Next, the second and third numbers are passed and swapped if the comparison returns 1. The same process repeats so that the array elements are arranged in ascending order.

</> Open Compiler

```
<?php
   function mysort($a, $b) {
      if ($a == $b) {
         return 0;
      }
```

```
      return ($a < $b) ? -1 : 1;
   }

   $a = array(3, 2, 5, 6, 1);

   usort($a, "mysort");

   foreach ($a as $key => $value) {
      echo "$key: $value\n";
   }
?>
```

It will produce the following **output** –

```
0: 1
1: 2
2: 3
3: 5
4: 6
```

Pass Callback to User-defined Function

Apart from the above built-in functions, you can define your own function that accepts one of the arguments as another function.

In the example below, we have two functions, **square()** and **cube()**, that return the square and cube of a given number.

Next, there is **myfunction()**, whose first argument is used as a variable function and the second argument to **myfunction()** is passed to it.

Thus, myfunction() internally calls square() or cube() to return either square or cube of a given number.

Example

</> Open Compiler

```
<?php
   function myfunction($function, $number) {
      $result = $function($number);
      return $result;
```

```
    }

    function cube($number) {
       return $number ** 2;
    }

    function square($number) {
       return $number ** 3;
    }

    $x = 5;

    $cube = myfunction('cube', $x);
    $square = myfunction('square', $x);

    echo "Square of $x = $square" . PHP_EOL;
    echo "Cube of $x = $cube" . PHP_EOL;
?>
```

It will produce the following **output** –

```
Square of 5 = 125
Cube of 5 = 25
```