PHP - Arrays

An array is a data structure that stores one or more data values having some relation among them, in a single variable. For example, if you want to store the marks of 10 students in a class, then instead of defining 10 different variables, it’s easy to define an array of 10 length.

Arrays in PHP behave a little differently than the arrays in C, as PHP is a dynamically typed language as against C which is a statically type language.

- An array in PHP is an ordered map that associates values to keys.
- A PHP array can be used to implement different data structures such as a stack, queue, list (vector), hash table, dictionary, etc.
- The value part of an array element can be other arrays. This fact can be used to implement tree data structure and multidimensional arrays.

There are two ways to declare an array in PHP. One is to use the built-in array() function, and the other is to use a shorter syntax where the array elements are put inside square brackets.

The array() Function

The built-in array() function uses the parameters given to it and returns an object of array type. One or more comma-separated parameters are the elements in the array.

```
array(mixed ...$values): array
```

Each value in the parenthesis may be either a singular value (it may be a number, string, any object or even another array), or a key-value pair. The association between the key and its value is denoted by the "=>" symbol.

Examples

```
$arr1 = array(10, "asd", 1.55, true);
$arr2 = array("one"=>1, "two"=>2, "three"=>3);
$arr3 = array(
   array(10, 20, 30),
   array("Ten", "Twenty", "Thirty"),
    array("physics"=>70, "chemistry"=>80, "maths"=>90)
);
```

Using Square Brackets [ ]

Instead of the array() function, the comma-separated array elements may also be put inside the square brackets to declare an array object. In this case too, the elements may be singular values or a string or another array.

```
$arr1 = [10, "asd", 1.55, true];
$arr2 = ["one"=>1, "two"=>2, "three"=>3];
$arr3 = [ [10, 20, 30],
   ["Ten", "Twenty", "Thirty"],
   ["physics"=>70, "chemistry"=>80, "maths"=>90] ];
```

Types of Arrays in PHP

There are three different kind of arrays and each array value is accessed using an ID which is called the array index.

- **Indexed Array** − An array which is a collection of values only is called an indexed array. Each value is identified by a positional index staring from "0". Values are stored and accessed in linear fashion.
- **Associative Array** − If the array is a collection of key-value pairs, it is called as an associative array. The key component of the pair can be a number or a string, whereas the value part can be of any type. Associative arrays store the element values in association with key values rather than in a strict linear index order.
- **Multi Dimensional Array** − If each value in either an indexed array or an associative array is an array itself, it is called a multi dimensional array. Values are accessed using multiple indices

NOTE − Built-in array functions is given in function reference PHP Array Functions

It may be noted that PHP internally considers any of the above types as an associative array itself. In case of an indexed array, where each value has index, the index itself is its key. The var_dump() function reveals this fact.

Example

In this example, **arr1** is an indexed array. However, var_dump()which displays the structured information of any object, shows that each value is having its index as its key.

</> Open Compiler

```
<?php
   $arr1 = [10, "asd", 1.55, true];
   var_dump($arr1);
?>
```

It will produce the following **output** –

```
array(4) {
  [0]=>
  int(10)
  [1]=>
  string(3) "asd"
  [2]=>
  float(1.55)
  [3]=>
  bool(true)
}
```

Example

The same principle applies to a multi-dimensional index array, where each value in an array is another array.

</> Open Compiler

```
<?php
   $arr1 = [
      [10, 20, 30],
      ["Ten", "Twenty", "Thirty"],
      [1.1, 2.2, 3.3]
   ];
   var_dump($arr1);
?>
```

It will produce the following **output** −

```
array(3) {
  [0]=>
  array(3) {
    [0]=>
    int(10)
    [1]=>
    int(20)
    [2]=>
    int(30)
  }
  [1]=>
  array(3) {
    [0]=>
    string(3) "Ten"
    [1]=>
    string(6) "Twenty"
    [2]=>
    string(6) "Thirty"
  }
  [2]=>
  array(3) {
    [0]=>
    float(1.1)
    [1]=>
    float(2.2)
    [2]=>
    float(3.3)
  }
}
```

Accessing the Array Elements

To access any element from a given array, you can use the array[key] syntax.

Example

For an indexed array, put the index inside the square bracket, as the index itself is anyway the key.

```
<?php
   $arr1 = [10, 20, 30];
   $arr2 = array("one"=>1, "two"=>2, "three"=>3);

   var_dump($arr1[1]);
   var_dump($arr2["two"]);
?>
```

It will produce the following **output** −

```
int(20)
int(2)
```

We shall explore the types of PHP arrays in more details in the subsequent chapters.