

Programming Basics

Vivify Academy



Arrays



What is an array?

In programming, an array is a group of related data values (called elements) that are grouped together in particular order.

An array is a data structure that stores one or more values in a single variable. For example if you want to store 100 numbers then instead of defining 100 different variables, it's easier to define an array with 100 elements. Array elements can be any type of data, including other arrays.

Arrays in PHP

There are two different kinds of arrays and each array value is accessed using an ID, which is called array index.

- **Numeric array** – An array with a numeric index. Values are stored and accessed using numbers in a linear fashion, starting with index 0.
- **Associative array** – An array with strings as indexes. This stores element values in association with key values rather than in a strict linear index order.

Aside from this, arrays can also be **multidimensional** – arrays containing one or more arrays. Values can be accessed using multiple indexes, which can be numbers or strings.

Numeric Arrays

<?php

```
// first method to create array:
```

```
// old syntax:
```

```
$numbers = array(1, 2, 3, 4, 5);
```

```
// new syntax:
```

```
$numbers = [1, 2, 3, 4, 5];
```

```
echo "<pre>";
```

```
var_dump($numbers);
```

```
echo "</pre>";
```

```
// second method to create array
```

```
$strings[0] = "one";
```

```
$strings[1] = "two";
```

```
$strings[2] = "three";
```

```
$strings[3] = "four";
```

```
$strings[4] = "five";
```

```
echo "<pre>";
```

```
var_dump($strings);
```

```
echo "</pre>";
```

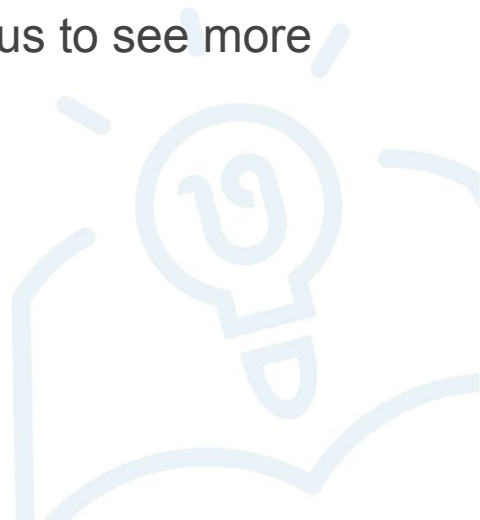
These arrays can store numbers, strings and any object but their index will be represented by numbers. By default array index starts from zero.

This example shows us how to create numeric arrays.

Why are we using `echo "<pre>"`?

The `var_dump` function will output the contents of an array correctly, but since HTML mostly ignores extra whitespace, everything may be printed on a single line and be difficult to read and understand.

The `<pre>` tag in HTML (short for *preformatted*) will show the text inside that element exactly as it was written in the document, which allows us to see more easily what complex data (such as arrays) looks like.



Moving through an array with **for**

```
<?php
```

```
$numbers = [6, 5, 4, 3];
```

```
for ($index = 0; $index < 4; $index++) {  
    echo $numbers[$index];  
}
```

```
?>
```



Associative Arrays

```
<?php
```

```
// first method to create associative array
$salaries = [
    "eric" => 2000,
    "carl" => 1000,
    "maceo" => 500,
];
```

```
var_dump($salaries);
```

```
// second method to create associative array
$salaries['eric'] = "high";
$salaries['carl'] = "medium";
$salaries['maceo'] = "low";
```

```
var_dump($salaries);
```

The associative arrays are very similar to numeric arrays in term of functionality but they use a different type of index.

Associative array will have their index as string so that you can establish a strong association between key and values.

To store the salaries of employees in an array, a numerically indexed array would not be the best choice. Instead, we could use the employees names as the keys in our associative array, and the value would be their respective salary.

The foreach loop statement


The foreach statement is used to loop through arrays. For each pass the value of the current array element is assigned to \$value automatically. We don't need to think about which index we're accessing, the program will “walk” through the array and keep track of our current position in it. We can also access the key if we need it, but for numeric arrays this is rarely needed.

Pseudocode

```
foreach (array as element) {  
    code to be executed;  
}
```

PHP

```
<?php  
  
$array = [1, 2, 3, 4, 5];  
  
foreach ($array as $element) {  
    echo "Value is $element <br>";  
}  
  
?>
```



Numeric Arrays

```
<?php
```

```
$numbers = [1, 2, 3, 4, 5];
```

```
// accessing (indexing)
```

```
var_dump($numbers[0]);
```

```
var_dump($numbers[2]);
```

```
var_dump($numbers[4]);
```

```
// iterating
```

```
foreach ($numbers as $number) {
```

```
    var_dump($number);
```

```
}
```

```
?>
```

This example is showing how to access value inside an array and how to iterate through a numeric array.

Associative Arrays

```
<?php
```

```
$salaries = [  
    "eric"   => 2000,  
    "carl"   => 1000,  
    "maceo"  => 500,  
];  
  
// accessing (indexing)  
var_dump($salaries['eric']);  
var_dump($salaries['carl']);  
var_dump($salaries['maceo']);  
  
// iterating  
foreach ($salaries as $name => $salary) {  
    echo "$name's salary is $salary. <br>";  
}
```

```
?>
```

This example is showing how to access value inside an array and how to iterate through associative array. Note the **\$key => \$value** syntax in the **foreach** loop; this way we can access the current index as well as the value.

This is not limited to associative arrays - it can be used for numeric arrays too, and associative arrays can be iterated without reading the **\$key**.

Multidimensional Arrays

```
<?php
```

```
$board = [  
    [ 'X', ' ', 'X' ],  
    [ 'X', 'O', 'O' ],  
    [ 'O', ' ', 'X' ],  
];
```

```
// accessing elements  
$board[0][2] === 'X';  
$board[1][1] === 'O';
```

```
?>
```

In a multi-dimensional array, also called a matrix, each element in the main array can also be an array. And each element in the sub-array can be an array, and so on. Values in a multi-dimensional array are accessed using multiple indexes.

In this example we create a two dimensional (3 by 3 elements) array to store a tic-tac-toe game board. This is an example of a multi-dimensional array with *numeric* indexes. The values are strings, but like we said before, they can be any type of data.

Multidimensional Arrays

<?php

```
$marks = [
    "eric" => [
        "physics"    => 5,
        "maths"      => 4,
    ],
    "carl" => [
        "physics"    => 3,
        "maths"      => 5,
    ]
];

// accessing multidimensional array values
echo "Marks for eric in physics : ";
echo $marks['eric']['physics'];
echo "<br>";

echo "Marks for carl in maths : ";
echo $marks['carl']['maths'];
echo "<br>";
```

In this example we create a two dimensional array to store marks of two students in two subjects. This is an example of a multi-dimensional *associative* array, which means the indexes are strings. Values can be of any type - in this case they are numbers.

Adding items to arrays

```
<?php
```

```
    array_push($fruits, "apple", "raspberry");
```

```
    echo "<pre>";  
    var_dump($fruits);  
    echo "</pre>";
```

```
    $fruits[] = "cherry";
```

```
    echo "<pre>";  
    var_dump($fruits);  
    echo "</pre>";
```

```
?>
```

If you only need to add one element to the array it's better to use `$array[]` than `array_push()` .

`array_push()` will raise a warning if the first argument is not an array. This differs from the `$array[]` behaviour where a new array is created.

Deleting items from arrays

```
<?php
```

```
$a1 = ['a' => 10, 'b' => 22, 'c' => 31];  
unset($a1['b']);  
// ['a' => 10, 'c' => 31]
```

```
echo "<pre>";  
var_dump($a1);  
echo "</pre>";
```

```
$a2 = [10, 22, 31];  
unset($a2[1]);  
// [0 => 10, 2 => 31]  
// note the missing index 1
```

```
echo "<pre>";  
var_dump($a2);  
echo "</pre>";
```



Solutions for missing indexes

```
<?php
```

```
// solution 1 for numeric arrays
$a3 = [10, 22, 31];
unset($a3[1]);
// [0 => 10, 2 => 31]

$a3 = array_values($a3);
// [0 => 10, 1 => 31]
// index is now continuous

echo "<pre>";
var_dump($a3);
echo "</pre>";
```

```
?>
```

```
<?php
```

```
// solution 2 for numeric arrays
$a4 = [10, 22, 31];

array_splice($a4, 1, 0);
// [0 => 10, 1 => 31]
// index is now continuous

echo "<pre>";
var_dump($a4);
echo "</pre>";
```

```
?>
```



Useful links

- <http://php.net/manual/en/language.types.array.php>
- <http://php.net/manual/en/function.unset.php>
- <http://php.net/manual/en/function.array-values.php>
- <http://php.net/manual/en/function.array-splice.php>



Exercise!

