# PHP Arrays

In this tutorial you'll learn how to store multiple values in a single variable in PHP.

## What is PHP Arrays

Arrays are complex variables that allow us to store more than one value or a group of values under a single variable name. Let's suppose you want to store colors in your PHP script. Storing the colors one by one in a variable could look something like this:

**Example** **Run this code »**

```php
<?php
$color1 = "Red";
$color2 = "Green";
$color3 = "Blue";
?>
```

But what, if you want to store the states or city names of a country in variables and this time this not just three may be hundred. It is quite hard, boring, and bad idea to store each city name in a separate variable. And here array comes into play.

## Types of Arrays in PHP

There are three types of arrays that you can create. These are:

- **Indexed array** — An array with a numeric key.
- **Associative array** — An array where each key has its own specific value.
- **Multidimensional array** — An array containing one or more arrays within itself.

## Indexed Arrays

An indexed or numeric array stores each array element with a numeric index. The following examples shows two ways of creating an indexed array, the easiest way is:

**Example** **Run this code »**

```php
<?php
// Define an indexed array

```

```php
$colors = array("Red", "Green", "Blue");
?>
```

**Note:** In an indexed or numeric array, the indexes are automatically assigned and start with 0, and the values can be any data type.

This is equivalent to the following example, in which indexes are assigned manually:

**Example** **Run this code »**

```php
<?php
$colors[0] = "Red";
$colors[1] = "Green";
$colors[2] = "Blue";
?>
```

## Associative Arrays

In an associative array, the keys assigned to values can be arbitrary and user defined strings. In the following example the array uses keys instead of index numbers:

**Example** **Run this code »**

```php
<?php
// Define an associative array
$ages = array("Peter"=>22, "Clark"=>32, "John"=>28);
?>
```

The following example is equivalent to the previous example, but shows a different way of creating associative arrays:

**Example** **Run this code »**

```php
<?php
$ages["Peter"] = "22";
$ages["Clark"] = "32";
$ages["John"] = "28";
?>
```

# Multidimensional Arrays

The multidimensional array is an array in which each element can also be an array and each element in the sub-array can be an array or further contain array within itself and so on. An example of a multidimensional array will look something like this:

**Example** **Run this code »**

```php
<?php
// Define a multidimensional array
$contacts = array(
    array(
        "name" => "Peter Parker",
        "email" => "peterparker@mail.com",
    ),
    array(
        "name" => "Clark Kent",
        "email" => "clarkkent@mail.com",
    ),
    array(
        "name" => "Harry Potter",
        "email" => "harrypotter@mail.com",
    )
);
// Access nested value
echo "Peter Parker's Email-id is: " . $contacts[0]["email"];
?>
```

# Viewing Array Structure and Values

You can see the structure and values of any array by using one of two statements — `var_dump()` or `print_r()`. The `print_r()` statement, however, gives somewhat less information. Consider the following example:

**Example** **Run this code »**

```php
<?php
// Define array
$cities = array("London", "Paris", "New York");

// Display the cities array
```

```
print_r($cities);
?>
```

The `print_r()` statement gives the following output:

```
Array ( [0] => London [1] => Paris [2] => New York )
```

This output shows the key and the value for each element in the array. To get more information, use the following statement:

**Example** **Run this code »**

```
<?php
// Define array
$cities = array("London", "Paris", "New York");

// Display the cities array
var_dump($cities);
?>
```

This `var_dump()` statement gives the following output:

```
array(3){[0]=>string(6) "London" [1]=>string(5) "Paris" [2]=>string(8) "NewYork"}
```

This output shows the data type of each element, such as a string of 6 characters, in addition to the key and value. In the next chapter you will learn how to sort array elements.

You will learn how to loop through the values of an array in the later chapter.