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:

```
<!DOCTYPE html>
<html lang="en">
<head>
   <title>PHP Arrays</title>
</head>
<body>

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

echo $color1;
echo "<br>";

echo $color2;
echo "<br>";

echo $color3;
?>

</body>
</html>
```

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:

```
<!DOCTYPE html>
<html lang="en">
<head>
   <title>PHP Indexed Arrays</title>
</head>
<body>

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

// Printing array structure
print_r($colors);
?>

</body>
</html>
```

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:

```
<!DOCTYPE html>
<html lang="en">
<head>
   <title>PHP Indexed Arrays</title>
</head>
<body>

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

// Printing array structure
```

```
print_r($colors);
?>

</body>
</html>
```

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:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>PHP Associative Array</title>
</head>
<body>

<?php
$ages = array("Peter"=>22, "Clark"=>32, "John"=>28);

// Printing array structure
print_r($ages);
?>

</body>
</html>
```

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

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>PHP Associative Array</title>
</head>
```

```
<body>

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

// Printing array structure
print_r($ages);
?>

</body>
</html>
```

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:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>PHP Multidimensional Array</title>
</head>
<body>

<?php
// Define nested 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"];
?>

</body>
</html>
```

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:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>PHP View Array Structure</title>
</head>
<body>

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

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

```
</body>
</html>
```

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:

```
<!DOCTYPE html>
<html lang="en">
<head>
   <title>PHP Get Complete Array Information</title>
</head>
<body>

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

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

</body>
</html>
```

PHP in_array() Function

Definition and Usage

The in_array() function searches an array for a specific value.

Note: If the search parameter is a string and the type parameter is set to TRUE, the search is case-sensitive.

```
<!DOCTYPE html>
<html>
<body>

<?php
$people = array("Peter", "Joe", "Glenn", "Cleveland");

if (in_array("Glenn", $people))
  {
  echo "Match found";
  }
else
  {
  echo "Match not found";
  }
?>

</body>
</html>
```

Syntax

```
in_array(search, array, type)
```

Parameter Values

Parameter	Description
search	Required. Specifies the what to search for

array	Required. Specifies the array to search
type	Optional. If this parameter is set to TRUE, the in_array() function searches for the search-string and specific type in the array.

Ex

```
<!DOCTYPE html>
<html>
<body>

<?php
$people = array("Peter", "Joe", "Glenn", "Cleveland", 23);

if (in_array("23", $people, TRUE))
  {
  echo "Match found<br>";
  }
else
  {
  echo "Match not found<br>";
  }
if (in_array("Glenn",$people, TRUE))
  {
  echo "Match found<br>";
  }
else
  {
  echo "Match not found<br>";
  }

if (in_array(23,$people, TRUE))
```

```
 {
 echo "Match found<br>";
 }
else
 {
 echo "Match not found<br>";
 }
?>

</body>
</html>
```

EX

```
<!DOCTYPE html>
<html>
<body>
<?php
$a = 0;
$output = false;
$inputs = array(6743478, 74698793,87894379, "Siva",
"sdjkj");
for ($a = 0; $a <= in_array("12345", $inputs); $a++) {
echo "The number is: $a <br>";
if(in_array("Siva", $inputs))
{
echo "The user definedvalue is found<br></br>";
}
else
{
echo "The user definedvalue is not found<br></br>";
}
}
?>
</body>
</html>
```

PHP Append Array

Syntax #1:

```
array_merge($array1, $array2);
```

Description – array_merge() is a PHP language built-in function. $array1 and $array2 are the two arrays we are looking to merge. It combined two single arrays into a single one.

Syntax #2:

```
array_push($array1, $array2);
```

Description – Again, array_push() is a PHP language built-in function. $array1 and $array2 are the two arrays we are looking to merge. In this process, the next array will appear in the next position of the first position. For example, if $array1 has the 5 elements, in this case, the complete $array2 will be placed on the 6th position.

Syntax #3:

```
array_push($array1, $val);
```

Description – array_merge() can be also used to adding the element to the array as well. $array1 is an array of whether $val is the value we want to add

in the $array1. In this process, the $val will be added as normal value to the $array1 next coming position.

Elements of the single associate array can be performed using array merge. This key-value pared array will be converted into a single array. There are various other ways we can perform the array_merge() functionalities.

Syntax #4:

```
array_combine($array1, $array2)
```

array_combine() can be used to combine two single arrays into the array of associate (into key-value pared) array.

Example #1 – Merge two arrays to form a single one

```
<?php

$array1 = array(1, 2, 3, 4);

$array2 = array(4,5, 6);

$arr_merge = array_merge($array1, $array2);
```

```
print_r($arr_merge);

?>
```

Example #2 – Merge two array using array_push

```
<?php

$array1 = array(1, 2, 3, 4);

$array2 = array(4,5, 6);

array_push($array1, $array2);

print_r($array1);

?>
```

Example #3 – Append a single element to an array

```
<?php

$array1 = array(1, 2, 3, 4);

array_push($array1, 2000);

print_r($array1);

?>
```

Example #4 – Feel the array by running loop

```
<?php

$testing = array();

for ($i=1; $i < 11 ; $i++) {

array_push($testing,$i);

}
```

```
print_r($testing);

?>
```

Example #5 – Merging associate array in PHP

```
<?php

$array1 = array("1" => "First");

$array2 = array("a2" => "Second", "a3" => "Third");

$result = array_merge($array1, $array2);

print_r($result);

?>
```

As we can see in this example, if we have the numeric key it will start from its traditional position. For the remaining, it will be added to the key-value.

Example #6 – Merging a single associate array in PHP

```
<?php

$array1 = array(1 => "Red", 3=>"Green", 2=>"Blue");

$result = array_merge($array1);

print_r($result);

?>
```

PHP | Delete an element from an array using unset() function

unset() function

To **remove an element from an array**, we can use a **PHP library unset() function**, it accepts the index and removes the element exists on the specified index.

We are also using another function **var_dump()** – which dumps the variable details i.e. here, it will print the array variable.

PHP code to remove an element from an array

```
<?php
//PHP code to remove an element from an array

//declaring an array of strings
$array = array('the','quick','brown','fox');
```

```php
//printing the array variable
var_dump($array);

//removing element from 1st index
unset ($array[1]);

//again, printing the array variable
var_dump($array);

//assigning the array after removing its element
//from 1st index to the new array
$array_new=array_values($array);

//printing the new array variable
var_dump($array_new);
?>
```

Output

```
array(4) {
  [0]=>
  string(3) "the"
  [1]=>
  string(5) "quick"
  [2]=>
  string(5) "brown"
  [3]=>
  string(3) "fox"
}
array(3) {
```

```
  [0]=>
  string(3) "the"
  [2]=>
  string(5) "brown"
  [3]=>
  string(3) "fox"
}
array(3) {
  [0]=>
  string(3) "the"
  [1]=>
  string(5) "brown"
  [2]=>
  string(3) "fox"
}
```

Explanation:

Here, We've created an array (`$array`) and then used the **PHP unset() method** to remove index 1 (which is the 2nd value since array starts from 0). Once that's removed, we print the array using `var_dump` but there is a problem that the indexes haven't updated. So, we create `$array_new` by using `array_values()` method on the existing `$array`.

PHP | Find the occurrences of a given element in an array

Given an array and an element, we have to find the occurrences of the element in the array.

To **find the occurrences of a given element in an array**, we can use two functions,

1. **array_keys()**
   It returns an array containing the specified keys (values).
2. **count()**
   It returns the total number of elements of an array i.e. count of the elements.

Firstly, we will call **array_keys()** function with the input array (in which we have to count the occurrences of the specified element) and element, **array_keys()** will return an array containing the keys, then we will call the function **count()** and pass the result of the **array_keys()** which will finally return the total number of occurrences of that element.

PHP code to find the occurrences of a given element

```
<?php
//an array of the string elements
$array = array(
    "The",
    "Quick",
    "Brown",
    "Fox",
    "Jumps",
    "Right",
    "Over",
    "The",
    "Lazy",
    "Dog"
    );


//word/element to find the the array
$word= "The";
//finding the occurances
$wordCount=count(array_keys($array, $word));
//printing the result
```

```
echo "Word <b>" .$word ."</b> Appears " .$wordCount ." time(s) in array\n";


//word/element to find the the array

$word= "Hello";

//finding the occurances

$wordCount=count(array_keys($array, $word));

//printing the result

echo "Word <b>" .$word ."</b> Appears " .$wordCount ." time(s) in array\n";

?>
```

Output

```
Word The Appears 2 time(s) in array

Word Hello Appears 0 time(s) in array
```

Explanation:

Here, we have an array `$array` with the string elements and then assigning the element to `$word` to find its occurrences, as mentioned the above example, element **"The"** appears two times in the `$array` and element **"Hello"** doesn't appear.

<!DOCTYPE html>

<html>

<body>

```
<?php

$arr = array('Hello','World!','Beautiful','Day!');

echo implode(" ",$arr)."<br>";

echo implode("+",$arr)."<br>";

echo implode("-",$arr)."<br>";

echo implode("X",$arr);

?>



</body>

</html>
```

EX

```
<html>

<body bgcolor="pink">

<h3>Explode Function</h3>
```

```
<?php
$str="I am simple boy!";
print_r(explode(" ",$str));
?>
</body>
</html>
```

EX

Ex

```
<html>
<body bgcolor="pink">
<h3>Implode Function</h3>
<?php
$arr=array ('I','am','simple','boy!');
echo implode(" ",$arr);
?>
</body>
</html>
```

Ex

```
<html>
<body bgcolor="pink">
```

```
<h3>Implode Function</h3>

<?php

$arr = array ('I','am','simple','boy!');

$space_separated = implode(" ", $arr);

$comma_separated = implode(" , ", $arr);

$slash_separated = implode(" / ", $arr);

$dot_separated = implode(" . ", $arr);

$hyphen_separated = implode(" - ", $arr);

echo $space_separated.'<br>';

echo $comma_separated.'<br>';

echo $slash_separated.'<br>';

echo $dot_separated.'<br>';

echo $hyphen_separated;

?>

</body>

</html>
```

EX

<!DOCTYPE html>

<html>

<body>

```
<?php

$str = 'one,two,three,four';



// zero limit

print_r(explode(',',$str,0));

print "<br>";



// positive limit

print_r(explode(',',$str,2));

print "<br>";



// negative limit

print_r(explode(',',$str,-1));

?>



</body>
```

</html>