PHP - Indexed Array

In PHP, the array elements may be a collection of key-value pairs or it may contain values only. If the array consists of values only, it is said to be an indexed array, as each element is identified by an incrementing index, starting with "0".

An indexed array in PHP may be created either by using the array() function or with the square bracket syntax.

```
$arr1 = array("a", 10, 9.99, true);
$arr2 = ["a", 10, 9.99, true];
```

Each element in the array has a positional index, the first element being at index "0". The var_dump() function reveals the structured information of these arrays as –

```
array(4) {
  [0]=>
  string(1) "a"
  [1]=>
  int(10)
  [2]=>
  float(9.99)
  [3]=>
  bool(true)
}
```

We can use the index to traverse the array, fetch the value at a given index or modify the value of an element in place.

Traversing an Indexed Array in PHP

Any type of PHP loop can be employed to traverse an array. If we want to use a **for** or **while** loop, we have to find the number of elements in the array with count() function and use its value as the test condition for the counted **for** or **while** loop.

Example

The following code uses a **for** loop to list all the elements in an indexed array.

```php
<?php
   $numbers = array(10, 20, 30, 40, 50);

   for ($i=0; $i<count($numbers); $i++){
      echo "numbers[$i] = $numbers[$i] \n";
   }
?>
```

It will produce the following **output** –

```
numbers[0] = 10
numbers[1] = 20
numbers[2] = 30
numbers[3] = 40
numbers[4] = 50
```

You can also use a **while** or **do-while** loop to traverse an indexed array. Here too, we need to find the array length with count() function.

Example

The following code traverses the given indexed array in reverse order –

</> Open Compiler

```php
<?php
   $numbers = array(10, 20, 30, 40, 50);
   $i = count($numbers)-1;
   while ($i>=0){
      echo "numbers[$i] = $numbers[$i] \n";
      $i--;
   }
?>
```

It will produce the following **output** –

```
numbers[4] = 50
numbers[3] = 40
numbers[2] = 30
```

```
numbers[1] = 20
numbers[0] = 10
```

Accessing the Array Elements Using Index

You can access any value from an array using the **array[index]** syntax. The value at a specific index may be assigned with a new value. The array is thus modified in place.

Example

The following program fetches the values from an array **$arr1** and places them in **$arr2** in the reverse order. So the value at 0th position in $arr1 becomes the last value in $arr2.

</> Open Compiler

```
<?php
   $arr1 = array(10, 20, 30, 40, 50);
   $size = count($arr1);

   for ($i=0; $i<$size; $i++){
      $arr2[$size-$i-1] = $arr1[$i];
   }

   for ($i=0; $i<$size; $i++){
      echo "arr1[$i] = $$arr1[$i] arr2[$i] = $$arr2[$i] \n";
   }
?>
```

It will produce the following **output** –

```
arr1[0] = $10 arr2[0] = $50
arr1[1] = $20 arr2[1] = $40
arr1[2] = $30 arr2[2] = $30
arr1[3] = $40 arr2[3] = $20
arr1[4] = $50 arr2[4] = $10
```

Traversing an Indexed Array Using "foreach" Loop

You can also use the **foreach** loop to iterate through an indexed array. Take a look at the following **example** –

</> Open Compiler

```
<?php
   $arr1 = [10, 20, 30, 40, 50];
   foreach ($arr1 as $val){
      echo "$val \n";
   }
?>
```

It will produce the following **output** –

```
10
20
30
40
50
```

Note that PHP internally treats the indexed array as an associative array, with the index being treated as the key. This fact can be verified by the var_dump() output of the array.

Example

We can unpack each element of an indexed array in the key and value variables with **foreach** syntax –

</> Open Compiler

```
<?php
   $arr1 = [10, 20, 30, 40, 50];
   foreach ($arr1 as $key => $val){
      echo "arr1[$key] = $val \n";
   }
?>
```

It will produce the following **output** –

```
arr1[0] = 10
arr1[1] = 20
arr1[2] = 30
arr1[3] = 40
arr1[4] = 50
```

In PHP, an array may be a mix of only values and key-value pairs. PHP assigns the index only to the values without keys.

Example

In this example, PHP assigns incrementing index to the numbers, skipping the key-value pair appearing in it.

</> Open Compiler

```php
<?php
   $arr1 = [10, 20,
         "vals" => ["ten", "twenty"],
         30, 40, 50];

   var_dump($arr1);
?>
```

It will produce the following **output** –

```
array(6) {
  [0]=>
  int(10)
  [1]=>
  int(20)
  ["vals"]=>
  array(2) {
    [0]=>
    string(3) "ten"
    [1]=>
    string(6) "twenty"
  }
  [2]=>
  int(30)
  [3]=>
```

```
  int(40)
  [4]=>
  int(50)
}
```