PHP - Associative Array

If each element in a PHP array is a key-value pair, such an array is called an **associative array**. In this type of array, each value is identified by its associated key and not an index.

- Associative arrays are used to implement data structures such as dictionary, maps, trees, etc.
- In PHP, the "=>" symbol is used to establish association between a key and its value.

How to Declare an Associative Array in PHP?

Both the approaches of declaring an array – the array() function and the square bracket notation – can be used.

```
$arr1 = array(
   "Maharashtra"=>"Mumbai",
   "Telangana"=>"Hyderabad",
   "UP"=>"Lucknow",
   "Tamilnadu"=>"Chennai"
);

$arr2 = ["Maharashtra"=>"Mumbai",
   "Telangana"=>"Hyderabad",
   "UP"=>"Lucknow",
   "Tamilnadu"=>"Chennai"];
```

If we call the var_dump() function, both the above arrays will show the similar structure –

```
array(4) {
   ["Maharashtra"]=>
   string(6) "Mumbai"
   ["Telangana"]=>
   string(9) "Hyderabad
   ["UP"]=>
   string(7) "Lucknow"
   ["Tamilnadu"]=>
```

```
   string(7) "Chennai"
}
```

The **key** part of each element in an associative array can be any number (integer, float or Boolean), or a string. The **value** part can be of any type. However, the float key is cast to an integer. So, a Boolean true/false is used as "1" or "0" as the key.

Example

Take a look at the following example −

</> Open Compiler

```
<?php
   $arr1 = array(
      10=>"hello",
      5.75=>"world",
      -5=>"foo",
      false=>"bar"
   );
   var_dump($arr1);
?>
```

It will produce the following **output** −

```
array(4) {
  [10]=>
  string(5) "hello"
  [5]=>
  string(5) "world"
  [-5]=>
  string(3) "foo"
  [0]=>
  string(3) "bar"
}
```

Note that the key 5.75 gets rounded to 5, and the key "true" is reflected as "0". If the same key appears more than once in an array, the key-value pair that appears last will be retained, discarding the association of the key with earlier value.

PHP internally treats even an indexed array as an associative array, where the index is actually the key of the value. It means the value at the 0th index has a key equal to "0",

and so on. A var_dump() on an indexed array also brings out this characteristics of a PHP array.

Iterating a PHP Associative Array

A **foreach** loop is the easiest and ideal for iterating through an associative array, although any other type of loop can also be used with some maneuver.

Example

Let us look at the **foreach** loop implementation, with each key value pair unpacked in two variables.

</> Open Compiler

```
<?php
   $capitals = array(
      "Maharashtra"=>"Mumbai",
      "Telangana"=>"Hyderabad",
      "UP"=>"Lucknow",
      "Tamilnadu"=>"Chennai"
   );

   foreach ($capitals as $k=>$v) {
      echo "Capital of $k is $v \n";
   }
?>
```

It will produce the following **output** −

```
Capital of Maharashtra is Mumbai
Capital of Telangana is Hyderabad
Capital of UP is Lucknow
Capital of Tamilnadu is Chennai
```

There is another way of using the **foreach** loop in PHP, where each element is stored in a variable. We can then separate the key and value parts using **array_search()** and use them in the loop body.

</> Open Compiler

```php
<?php
   $capitals = array(
      "Maharashtra"=>"Mumbai",
      "Telangana"=>"Hyderabad",
      "UP"=>"Lucknow",
      "Tamilnadu"=>"Chennai"
   );

   foreach ($capitals as $pair) {
      $cap = array_search($pair, $capitals);
      echo "Capital of $cap is $capitals[$cap] \n";
   }
?>
```

It will produce the following **output** –

```
Capital of Maharashtra is Mumbai
Capital of Telangana is Hyderabad
Capital of UP is Lucknow
Capital of Tamilnadu is Chennai
```

To use for, **while** or **do-while** loop, we have to first get the array of all the keys (use array_keys()), find the size and use it as the test condition in the loop syntax.

Example

Here is how we can use a **for** loop to traverse an associative array –

</> Open Compiler

```php
<?php
   $capitals = array(
      "Maharashtra"=>"Mumbai",
      "Telangana"=>"Hyderabad",
      "UP"=>"Lucknow",
      "Tamilnadu"=>"Chennai"
   );
   $keys=array_keys($capitals);

   for ($i=0; $i<count($keys); $i++){
      $cap = $keys[$i];
```

```
      echo "Capital of $cap is $capitals[$cap] \n";
   }
?>
```

It will produce the following **output** –

```
Capital of Maharashtra is Mumbai
Capital of Telangana is Hyderabad
Capital of UP is Lucknow
Capital of Tamilnadu is Chennai
```


Accessing the Value with its Key

In an associative array, the key is the identifier of value instead of index. Hence, to fetch value associated with a certain key, use $arr[key] syntax. The same can be used to update the value of a certain key.

Example

In the following code, an associative array **$arr1** is declared. Another array **$arr2** is created such that it stores each pair from **$arr1** with the value of each key being doubled.

</> Open Compiler

```
<?php
   $arr1 = array("a"=>10, "b"=>20, "c"=>30, "d"=>40);
   foreach ($arr1 as $k=>$v){
      $arr2[$k] = $v*2;
   }
   print_r($arr2);
?>
```

It will produce the following **output** –

```
Array
(
  [a] => 20
```

```
    [b] => 40
    [c] => 60
    [d] => 80
)
```

The **print_r()** function used here displays the data stored in the array in an easy to understand human readable form.