Working with Array

Prepared by Deepali Sonawane, MIT-WPU, Pune

Array

- Array is a group of elements of **similar** data type.
- Array is a set of **homogeneous** elements.
- An array stores **multiple values** in one **single variable**.
- An array is a special variable, which can hold **more than one value** at a time.
- In PHP, the **array()** function is used to create an array.
- Syntax:

  $array_name = array();
- There are three types of array:
  1. Indexed array
  2. Associative array
  3. Multidimensional array

Indexed Array

- An indexed array with a **numeric index** where values are stored linearly.
- Example : $num = array(2,5,8,15);

```
              $courses = array("OS", "Java", "PHP", "IOT");
Method 1:        for($i=0 ; $i <count($num) ; $i++)
                 {
                         echo "<br>",$num[$i];
                 }
Method 2:        foreach ($cources  as $value)
                 {
                         echo "$value <br>";
                 }
Method 3:        print_r($courses);
```

Associative Array

- The associative arrays are very similar to numeric arrays in term **of functionality** but they are different in terms of **their index**.
- Associative array can have their **index as string** so that we can establish a strong association between **key and values**.
- An associative array with a **string index** where instead of linear storage, each value can be assigned a specific key.
- This stores element values in association with key values rather than in a strict linear index order.
- Example : $name_percet = array(“Anil"=>"60”, “Monal"=>”70",

```
        "Ram"=>“65", “Sonal"=>”89”);
    foreach($name_percet  as $key => $value)
    {
            echo "Key=" , $key , ", Value=" , $value;
            echo "<br>";
    }
```

Multidimensional Array

- An array which contains single or multiple array within it and can be accessed via multiple indices.
- A multi-dimensional array each element in the main array can also be an array.
- Example : $student = array (

```
  array(“Amit",21,90),
  array(“Annad",22,45),
  array(“Rutuja",21,60),
  array(“Seema",22,55)
);
for ($i = 0; $i < 4; $i++)
{
  for ($j = 0; $j < 3; $j++)
   {
    echo $student[$i][$j];
   }
  echo "<br>";
}
```

Array Library Functions

- Sort functions for array
  1. sort(array) - sort arrays in ascending order
  2. rsort(array) - sort arrays in descending order
  3. asort(array) - sort associative arrays in ascending order, according to the value
  4. ksort(array) - sort associative arrays in ascending order, according to the key
  5. arsort(array) - sort associative arrays in descending order, according to the value
  6. krsort(array) - sort associative arrays in descending order, according to the key

Array Library Functions

1. count(array) : Returns number of elements of specified array.
2. sizeof(array) : Returns the number of elements in an array.
3. array_change_key_case(*array*, *case*) : Changes all keys in an array to lowercase or uppercase. Value of case is CASE_LOWER, CASE_UPPER
4. array_chunk(*array*, *size*) : Splits an array into chunks of new arrays.
5. array_combine(*keys_array*, *values_array*) : Creates an array by using the elements from one "keys" array and one "values" array.
6. array_count_values(array) : Occurrence of all the values of an array.
7. array_diff(*array1, array2, array3, ...*) : Compares the values of two (or more) arrays, and returns the differences.
8. array_flip(array) : Exchanges all keys with their associated values in an array.
9. array_merge(*array1, array2, array3, ...*) : Merges one or more arrays into one array.

Array Library Functions

10. array_intersect(*array1, array2, array3, …*) : Compares the values of two (or more) arrays, and returns the matches.
11. array_key_exists(*key, array*) : Checks an array for a specified key, and returns true if the key exists and false if the key does not exist.
12. array_keys(*array*) : Returns an array containing the keys.
13. array_values(*array*) : Returns an array containing all the values of an array.
14. array_push(*array, value1, value2, …*) : Inserts one or more elements to the end of an array.
15. array_pop(*array*) : Deletes the last element of an array.
16. array_shift(*array*) : Deletes the first element from an array.
17. array_unshift(*array, value1, value2, value3, …*) : New array values will be inserted in the beginning of the array
18. array_product(*array*) : Calculates and returns the product of an array elements.
19. array_sum(*array*) : Calculates and returns the sum of an array elements.

Array Library Functions

20. array_replace(*array1, array2,….*) : Replaces the values of the first array with the values from following arrays.
21. array_reverse(*array[, preserve]*): Returns an array in the reverse order. Value of preserve is true or false and it specifies if the function should preserve or reset the keys. Default value of preserve is false.
22. array_search(*value, array[, strict]*) : Search an array for a value and returns the key.

    strict checks data type of value. Default value of strict is false.
20. array_slice(*array, start[, length, preserve]*) : Returns selected parts of an array.
21. array_splice(*array1, start, length, array2*) : Removes selected elements from an array1 and replaces it with new elements of array2.
22. array_unique(*array) :* Removes duplicate values from an array.
23. extract(*array) :* This function uses array keys as variable names and values as variable values.

Thank you