Unit-4 Array and Function

What is an Array?

- An array is a special variable, which can hold more than one value at a time.
- If you have a list of items (a list of car names, for example), storing the cars in single variables could look like this:

```
$cars1 = "Volvo";
$cars2 = "BMW";
$cars3 = "Toyota";
```

- An array can hold many values under a single name, and you can access the values by referring to an index number.

Create an Array in PHP

- In PHP, there are three types of arrays:
- **Indexed arrays** - Arrays with a numeric index
- **Associative arrays** - Arrays with named keys
- **Multidimensional arrays** - Arrays containing one or more arrays

PHP Indexed Arrays

- There are two ways to create indexed arrays:
- The index can be assigned automatically (index always starts at 0), like this:

```
$cars = array("Volvo", "BMW", "Toyota"); OR

$cars[0] = "Volvo";
$cars[1] = "BMW";
$cars[2] = "Toyota";
```

- The following example creates an indexed array named $cars, assigns three elements to it, and then prints a text containing the array values:

Example

```
<?php
$cars = array("Volvo", "BMW", "Toyota");
echo "I like " . $cars[0] . ", " . $cars[1] . " and " . $cars[2] . ".";
?>
```

Get The Length of an Array - The count() Function

- The count() function is used to return the length (the number of elements) of an array:

Example

```
<?php
$cars = array("Volvo", "BMW", "Toyota");
echo count($cars);
?>
```

Loop Through an Indexed Array

- To loop through and print all the values of an indexed array:

Example

```
<?php
$cars = array("Volvo", "BMW", "Toyota");
$arrlength = count($cars);

for($x = 0; $x < $arrlength; $x++) {
    echo $cars[$x];
    echo "<br>";
}
?>
```

PHP Associative Arrays

- Associative arrays are arrays that use named keys that you assign to them.
- There are two ways to create an associative array:

```
$age = array("Peter"=>"35", "Ben"=>"37",
"Joe"=>"43");
```

OR

```
$age['Peter'] = "35";
$age['Ben'] = "37";
$age['Joe'] = "43";
```

- The named keys can then be used in a script:

Example

```
<?php
$age
= array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");
echo "Peter is " . $age['Peter'] . " years old.";
?>
```

Loop Through an Associative Array

- To loop through and print all the values of an associative array

Example

```
<?php
$age
= array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");

   foreach($age as $x => $x_value) {
    echo "Key=" . $x . ", Value=" . $x_value;
    echo "<br>";
}
?>
```

Functions

- count() :Return the number of elements in an array:

```
<?php
$cars=array("Volvo","BMW","Toyota");
echo count($cars);
?>
```

- **Definition and Usage**

  The count() function returns the number of elements in an array.

PHP current() Function

- The value of the current element in an array:

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

echo current($people) . "<br>";
?>
```

- **Definition and Usage**
- The current() function returns the value of the current element in an array.
- Every array has an internal pointer to its "current" element, which is initialized to the first element inserted into the array.

PHP end() Function

- The value of the current and the last element in an array:

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

echo current($people) . "<br>";
echo end($people);
?>
```

- **Definition and Usage**
- The end() function moves the internal pointer to, and outputs, the last element in the array.

PHP next() Function

- The value of the current and the next element in the array:

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

echo current($people) . "<br>";
echo next($people);
?>
```

- **Definition and Usage**
- The next() function moves the internal pointer to, and outputs, the next element in the array.

PHP prev() Function

- The value of the current, next and previous element in the array:

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

echo current($people) . "<br>";
echo next($people) . "<br>";
echo prev($people);
?>
```

- **Definition and Usage**
- The prev() function moves the internal pointer to, and outputs, the previous element in the array.

PHP sort() Function

- Sort the elements of the $cars array in ascending alphabetical order:

```
<?php
$cars=array("Volvo","BMW","Toyota");
sort($cars);

$clength=count($cars);
for($x=0;$x<$clength;$x++)
  {
  echo $cars[$x];
  echo "<br>";
  }
?>
```

- **Definition and Usage**
- The sort() function sorts an indexed array in ascending order.

PHP asort() Function

- Sort an associative array in ascending order, according to the value:

```
<?php
$age=array("Peter"=>"35","Ben"=>"37","Joe"=>"43");
asort($age);

foreach($age as $x=>$x_value)
   {
   echo "Key=" . $x . ", Value=" . $x_value;
   echo "<br>";
   }
?>
```

- **Definition and Usage**
- The asort() function sorts an associative array in ascending order, according to the value.

PHP rsort() Function

- Sort the elements of the $cars array in descending alphabetical order:

```
<?php
$cars=array("Volvo","BMW","Toyota");
rsort($cars);

$clength=count($cars);
for($x=0;$x<$clength;$x++)
  {
  echo $cars[$x];
  echo "<br>";
  }
?>
```

- **Definition and Usage**
- The rsort() function sorts an indexed array in descending order.

PHP arsort() Function

- Sort an associative array in descending order, according to the value:

```
<?php
$age=array("Peter"=>"35","Ben"=>"37","Joe"=>"43");
arsort($age);

foreach($age as $x=>$x_value)
   {
   echo "Key=" . $x . ", Value=" . $x_value;
   echo "<br>";
   }
?>
```

- **Definition and Usage**
- The arsort() function sorts an associative array in descending order, according to the value.

PHP array_merge() Function

- Merge two arrays into one array:

```
<?php
$a1=array("red","green");
$a2=array("blue","yellow");
print_r(array_merge($a1,$a2));
?>
```

- **Definition and Usage**
- The array_merge() function merges one or more arrays into one array.

PHP array_reverse() Function

- Return an array in the reverse order:

```
<?php
$a=array("a"=>"Volvo","b"=>"BMW","c"=>"
Toyota");
print_r(array_reverse($a));
?>
```

- **Definition and Usage**
- The array_reverse() function returns an array in the reverse order.

PHP array_diff() Function

- Compare the **values** of two arrays, and return the differences:

```
<?php
$a1=array("a"=>"red","b"=>"green","c"=>"blue","d"=>"yellow");
$a2=array("e"=>"red","f"=>"green","g"=>"blue");

$result=array_diff($a1,$a2);
print_r($result);
?>
```

- **Definition and Usage**
- The array_diff() function compares **the values** of two (or more) arrays, and returns the differences.
- This function compares the values of two (or more) arrays, and return an array that contains the entries from *array1* that are not present in *array2* or *array3*, etc.

PHP array_shift() Function

- Remove the first element (red) from an array, and return the value of the removed element:

```
<?php
$a=array("a"=>"red","b"=>"green","c"=>"blue");
echo array_shift($a)."<br>";
print_r ($a);
?>
```

- **Definition and Usage**
- The array_shift() function removes the first element from an array, and returns the value of the removed element.

PHP array_slice() Function

- Start the slice from the third array element, and return the rest of the elements in the array:

```
<?php
$a=array("red","green","blue","yellow","brown");
print_r(array_slice($a,2));
?>
```

- **Definition and Usage**
- The array_slice() function returns selected parts of an array.

PHP array_unique() Function

- Remove duplicate values from an array:

```
<?php
$a=array("a"=>"red","b"=>"green","c"=>"red");
print_r(array_unique($a));
?>
```

- **Definition and Usage**
- The array_unique() function removes duplicate values from an array. If two or more array values are the same, the first appearance will be kept and the other will be removed.
- **Note:** The returned array will keep the **first** array item's key type.

PHP array_unshift() Function

- Insert the element "blue" to an array:

```
<?php
$a=array("a"=>"red","b"=>"green");
array_unshift($a,"blue");
print_r($a);
?>
```

- **Definition and Usage**
- The array_unshift() function inserts new elements to an array. The new array values will be inserted in the beginning of the array.
- **Tip:** You can add one value, or as many as you like.
- **Note:** Numeric keys will start at 0 and increase by 1. String keys will remain the same.

PHP array_keys() function

- Return an array containing the keys:

```
<?php
$a=array("Volvo"=>"XC90","BMW"=>"X5",
"Toyota"=>"Highlander");
print_r(array_keys($a));
?>
```

- **Definition and Usage**
- The array_keys() function returns an array containing the keys.

PHP array_key_exists() Function

- Check if the key "Volvo" exists in an array:

```
<?php
$a=array("Volvo"=>"XC90","BMW"=>"X5");
if (array_key_exists("Volvo",$a))
  {
  echo "Key exists!";
  }
else
  {
  echo "Key does not exist!";
  }
?>
```

- **Definition and Usage**
- The array_key_exists() function checks an array for a specified key, and returns true if the key exists and false if the key does not exist.
- **Tip:** Remember that if you skip the key when you specify an array, an integer key is generated, starting at 0 and increases by 1 for each value. (See example 2)

PHP array_push() Function

- Insert "blue" and "yellow" to the end of an array:
- <?php
  $a=array("red","green");
  array_push($a,"blue","yellow");
  print_r($a);
  ?>
- **Definition and Usage**
- The array_push() function inserts one or more elements to the end of an array.
- **Tip:** You can add one value, or as many as you like.
- **Note:** Even if your array has string keys, your added elements will always have numeric keys (See example below).

PHP array_pop() Function

- Delete the last element of an array:

```
<?php
$a=array("red","green","blue");
array_pop($a);
print_r($a);
?>
```

- **Definition and Usage**
- The array_pop() function deletes the last element of an array.

PHP array_multisort() Function

- Return a sorted array in ascending order:

```
<?php
$a=array("Dog","Cat","Horse","Bear","Zebra");
array_multisort($a);
print_r($a);
?>
```

- **Definition and Usage**
- The array_multisort() function returns a sorted array. You can assign one or more arrays. The function sorts the first array, and the other arrays follow, then, if two or more values are the same, it sorts the next array, and so on.
- **Note:** String keys will be maintained, but numeric keys will be re-indexed, starting at 0 and increase by 1.
- **Note:** You can assign the sorting order and the sorting type parameters after each array. If not specified, each array parameter uses the default values.

PHP array_search() Function

- Search an array for the value "red" and return its key:

```
<?php
$a=array("a"=>"red","b"=>"green","c"=>"blu
e");
echo array_search("red",$a);
?>
```

- Definition and Usage
- The array_search() function search an array for a value and returns the key.

THANK YOU