Array functions in PHP

array():

PHP array() function creates and returns an array. It allows you to create indexed, associative and multidimensional arrays.

```
<?php
$season=array("summer","winter","spring","autumn");
echo "Season are: $season[0], $season[1], $season[2] and $season[3]";
?>
```

Output:

```
Season are: summer, winter, spring and autumn
```

Array_values():

Return all the values of an array (not the keys)

```
<?php
$a=array("Name"=>"abc       ","Age"=>"41","Country"=>"INDIA");
print_r(array_values($a));
?>
```

array_change_key_case() :

PHP array_change_key_case() function changes the case of all key of an array.

Note: It changes case of key only.

```
<?php
$salary=array("Sonoo"=>"550000","Vimal"=>"250000","Ratan"=>"200000");
print_r(array_change_key_case($salary,CASE_UPPER));
?>
```

Output:

```
Array ( [SONOO] => 550000 [VIMAL] => 250000 [RATAN] => 200000 )
```

array_chunk() :

PHP array_chunk() function splits array into chunks. By using array_chunk() method, you can divide array into many parts.

```
<?php
$salary=array("Sonoo"=>"550000","Vimal"=>"250000","Ratan"=>"200000");
print_r(array_chunk($salary,2));
?>
```

Output:

```
Array (
[0] => Array ( [0] => 550000 [1] => 250000 )
[1] => Array ( [0] => 200000 )
)
```

Count():

PHP count() function counts all elements in an array.

```
<?php
$season=array("summer","winter","spring","autumn");
echo count($season);
?>
```

Output :4

Sort()

PHP sort() function sorts all the elements in an array.

```
<?php
$season=array("summer","winter","spring","autumn");
sort($season);
foreach( $season as $s )
{
  echo "$s<br />";
}
?>
```

Output:

```
autumn
spring
summer
winter
```

array_reverse()

PHP array_reverse() function returns an array containing elements in reversed order.

```
<?php
```

```
$season=array("summer","winter","spring","autumn");
$reverseseason=array_reverse($season);
foreach( $reverseseason as $s )
{
  echo "$s<br />";
}
?>
```

Output:

```
autumn
spring
winter
summer
```

array_search():

PHP array_search() function searches the specified value in an array. It returns key if search is successful.

```
<?php
$season=array("summer","winter","spring","autumn");
$key=array_search("spring",$season);
echo $key;
?>
```

Output:2

asort():

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

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

Output:

```
Key=Peter, Value=35
Key=Ben, Value=37
Key=Joe, Value=43
```

Arsort():

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

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

Output:

Key=Joe, Value=43
Key=Ben, Value=37
Key=Peter, Value=35

Ksort():

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

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

Output:

```
Key=Ben, Value=37
Key=Joe, Value=43
Key=Peter, Value=35
```

Krsort():

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

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

Output:

```
Key=Peter, Value=35
Key=Joe, Value=43
Key=Ben, Value=37
```

Difference between = = and = = =

The comparison operator called Equal Operator is the double equal sign “==”. This operator accepts two inputs to compare and returns true value if both of the values are same (It compares only value of variable, not data types) and return a false value if both of the values are not same.

```
<?php

// Variable contains integer value
$x = 999;

// Vatiable contains string value
$y = '999';

// Compare $x and $y
if ($x == $y)
    echo 'Same content';
else
    echo 'Different content';
?>
```

Output:

```
Same Content
```

The comparison operator called as the Identical operator is the triple equal sign “===”. This operator allows for a **much stricter comparison between the given variables or values.** This operator returns true if both variable contains **same information and same data types** otherwise return false.

```
<?php

// Variable contains integer value
$x = 999;

// Vatiable contains string value
$y = '999';
```

```
// Compare $x and $y
if ($x === $y)
    echo 'Same content';
else
    echo ' Data type or value are different';
?>
```

Output:

```
Data type or value are different
```

Differentiate include and include_once:

The include_once statement includes and evaluates the specified file during the execution of the script. This is a behavior similar to the include statement, with the only difference being that if the code from a file has already been included, it will not be included again, and include_once returns true. As the name suggests, the file will be included just once.

Connection.php

```
$server="localhost";

$user="root";

$pwd="";

$db="ass1";

$conn=new mysqli();
```

Viewemp.php

```
<?php

Include_once  connection.php

$str="select * from emp";

$res=$conn->query($str);

?>

<table>

.
```

…

</table>

<input type=”submit” name=”btnsubmit”>

Single –Quoted string: It is the easiest way to define a string. You can use it when you want the string to be exactly as it is written. All the escape sequences like **\r** or **\n**, will be output as specified instead of having any special meaning. Single-quote is usually faster in some cases. The special case is that if you to display a literal single-quote, escape it with a **backslash (\)** and if you want to display a backslash, you can escape it with another **backslash (\\)**.

```
<?php

$a=10;

Echo ‘value of variable a is =$a’

?>
```

Output:

```
Value of variable is =$a
```

Double-quoted strings: By using Double quotes the PHP code is forced to evaluate the whole string. The main difference between double quotes and single quotes is that by using double quotes, you can include variables directly within the string. It interprets the Escape sequences. Each variable will be replaced by its value.That is variable is evaluated.

```
<?php

$a=10;

Echo “value of variable a is =$a”

?>
```

Output:

```
Value of variable is =10
```