String and Array in PHP

https://phpgurukul.com

Index

1.String

A string is a series of characters. There are exactly 256 different characters possible. The present stable versions of PHP – PHP4 and PHP5, have no native support for Unicode. There aren't string length limitations in PHP except the server's available memory and the configuration of the php.ini settings file.
The string, like a variable, will have to be created first. There are two ways to use a string in PHP – you can store it in a function or in a variable. In the example below, we will create a string twice – the first time storing it in a variable, and the second time – in a function, in our case – an echo

Basic Example

```
<?php

$myString = "This is a string!";

echo "This is a string!";

echo $myString;

?>
```

The output of this PHP file will be:

This is a string!This is a string!

String basics

1. Single quote strings (‘ ‘) –
Single quotes represent ‚simple strings,‘ where almost all characters are used literally
2. Double quote strings(” “) –
Complex strings‘ that allow for special escape sequences (for example, to insert special characters) and for variable substitution
Ex-

```
$a="PHPGURUKUL";

echo ‘Hello $a’;

echo "Hello $a n welcome";
```

Output

Hello $a
Hello PHPGURUKUL
welcome

Clearly, this “simple” syntax won’t work in those situations in which the name of the variable you want to interpolated is positioned in such a way inside the string that the parser wouldn’t be able to parse its name in the
way you intend it to. In these cases, you can encapsulate the variable’s name in braces.

```
$me = ’Davey’;

$names = array (’Anuj’, ’Sanjeev’, ’Rahul’);

echo "There cannot be more than two {$me}s!";

echo "Citation: {$names[1]}[1987]";
```

3. The Heredoc Syntax

Used to declare complex strings, the functionality it provides is similar to double quotes, with the exception that, because heredoc uses a special set of tokens to encapsulate the string, it’s easier to declare strings that include many double quote characters.

```
$who = "World";

echo <<<TEXT

So I said, "Hello $who"

TEXT;
```

Escaping Literal Values

• All three string-definition syntax feature a set of several characters that require escaping in order to be interpreted as literals.

```
echo 'This is 'my' string';

$a = 10;

echo "The value of $a is "$a".";

echo "Here's an escaped backslash:";
```

String as arrays

You can access the individual characters of a string as if they were members of an array.

```
$string = 'abcdef';

echo $string[1]; // Outputs 'b'
```

String comparison == and ===

```
$string = '123aa';

if ($string == 123) {

// The string equals 123

}
```

- You'd expect this comparison to return false, since the two operands are not the same. However, PHP first transparently converts the contents of $string to the integer 123, thus making the comparison true.
- Naturally, the best way to avoid this problem is to use the identity operator ===

2.String functions

String functions –strcmp(), strcasecmp()

strcmp(), **strcasecmp()** both returns zero if the two strings passed to the function are equal. These are identical, with the exception that the former is case-sensitive, while the latter is not.

```
$str = "Hello World";
if (strcmp($str, "hello world") === 0) {
// We won't get here, because of case sensitivity
}
if (strcasecmp($str, "hello world") === 0) {
// We will get here, because strcasecmp() is case-insensitive
}
```

String functions –strcasencmp()

strcasencmp() allows you to only test a given number of characters inside two strings.

```
$s1 = 'abcd1234';
$s2 = 'abcd5678';
// Compare the first four characters
echo strcasencmp ($s1, $s2, 4);
```

String functions –strlen()

strlen() is used to determine the length of a string.

```
$a="PHPGURUKUL Education Parnter";
echo strlen($a);
```

Output – 28

String functions –strtr()

strtr() used to translate certain characters of a string into other characters.

- Single character version

```
<?php
echo strstr ('abc', 'a', '1');
?>
```

- **Output –** 1bc

Multiple-character version

```
<?php
$subst = array ('1' => 'one','2' => 'two');
echo strtr ('123', $subst);
?>
```

Output- onetwo3

String functions –strpos()

- strpos() allows you to find the position of a substring inside a string. It returns either the numeric position of the substring's first occurrence within the string, or false if a match could not be found.
- You can also specify an optional third parameter to strpos() to indicate that you want the search to start from a specific position within the haystack.

```
<?php
$haystack = '123456123456';
$needle = '123';
echo strpos ($haystack, $needle);
echo strpos ($haystack, $needle, 1);
?>
```

Output –
0
6

String functions –stripos() , strrpos()

stripos() is case-insensitive version of strpos().

```
echo stripos('Hello World', 'hello');
```

Output- 0

Strpos() does the same as strpos(), but in the revers order.

```
echo strrpos ('123123', '123');
```

Output- 3

String functions –strstr()

The strstr() function works similarly to strpos() in that it searches the main string for a substring. The only real difference is that this function returns the portion of the main string that starts with the sub string instead of the latter's position:

```
$haystack = '123456';

$needle = '34';

echo strstr ($haystack, $needle);
```

Output- 3456

String functions –stristr()

stristr() is case-insensitive version of strstr().

```
echo stristr('Hello My World', 'my');
```

Output- My World

String functions –str_replace(), str_ireplace()

str_replace() used to replace portions of a string with a different substring.

```
echo str_replace("World", "Reader", "Hello World");
```

Output- Hello Reader

Str_ireplace() is the case insensitive version of str_replace().

```
echo str_ireplace("world", "Reader", "Hello World");
```

Output- Hello Reader

Optionally, you can specify a third parameter, that the function fills, upon return, with the number of substitutions made:

```
$a = 0;

str_replace (’a’, ’b’, ’a1a1a1’, $a);

echo $a;
```

Output- 3

If you need to search and replace more than one needle at a time, you can pass the first two arguments to str_replace() in the form of arrays.

```
echo str_replace(array("Hello", "World"), array("Bonjour", "Monde"), "HelloWorld");
```

Output – Hello Reader

```
echo str_replace(array("Hello", "World"), "Bye", "Hello World");
```

Output – Bye Bye

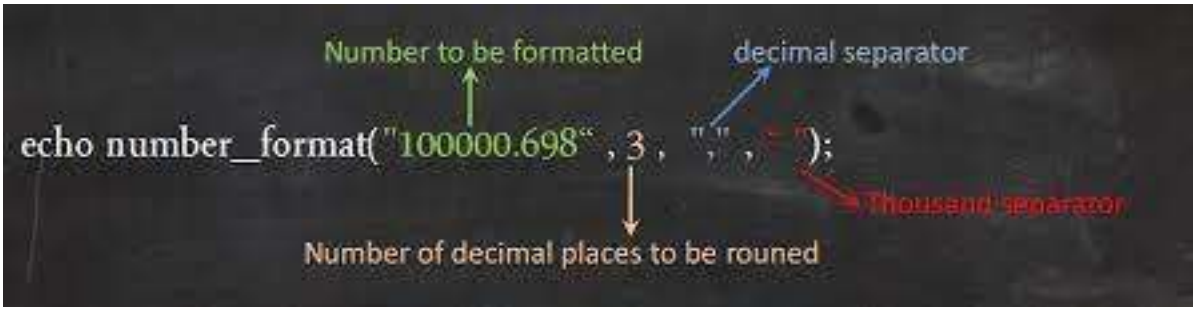
String functions –substr()

The very flexible and powerful substr() function allows you to extract a substring from a larger string.

```
echo substr ($x, 0, 3);// outputs 123

echo substr ($x, 1, 1);// outputs 2

echo substr ($x, -2); //outputs 67

echo substr ($x, 1); //outputs 234567

echo substr ($x, -2, 1); //outputs 6
```

String functions –number_format()

Number formatting is typically used when you wish to output a number and separate its digits into thousands and decimal points.



Output- 100 000,698

3.Array

An array is a group of the same data type which is used to store multiple values in a single variable. In an array each data item is referenced by its name and a number that is called a subscript. The elements of an array are accessed by specifying a subscript after the array variable name. The first element of an array has a subscript of 0, the next is 1, and so on. The last element has a subscript of n-1, where n is the number of elements.

Types of Array

There are three types of arrays in PHP, which are as follows:

- Numeric Index Array
- Associative Array
- Multidimensional Array

Numeric Index Array

A numeric index array stores each array element with a numeric index i.e. subscript. In a numeric indexed array, the keys are numeric and starts with subscript of 0, the next is 1, and so on. Numeric indexed arrays use integer/numbers as their index number to identify each item of the array.

Syntax of Numeric Index Array

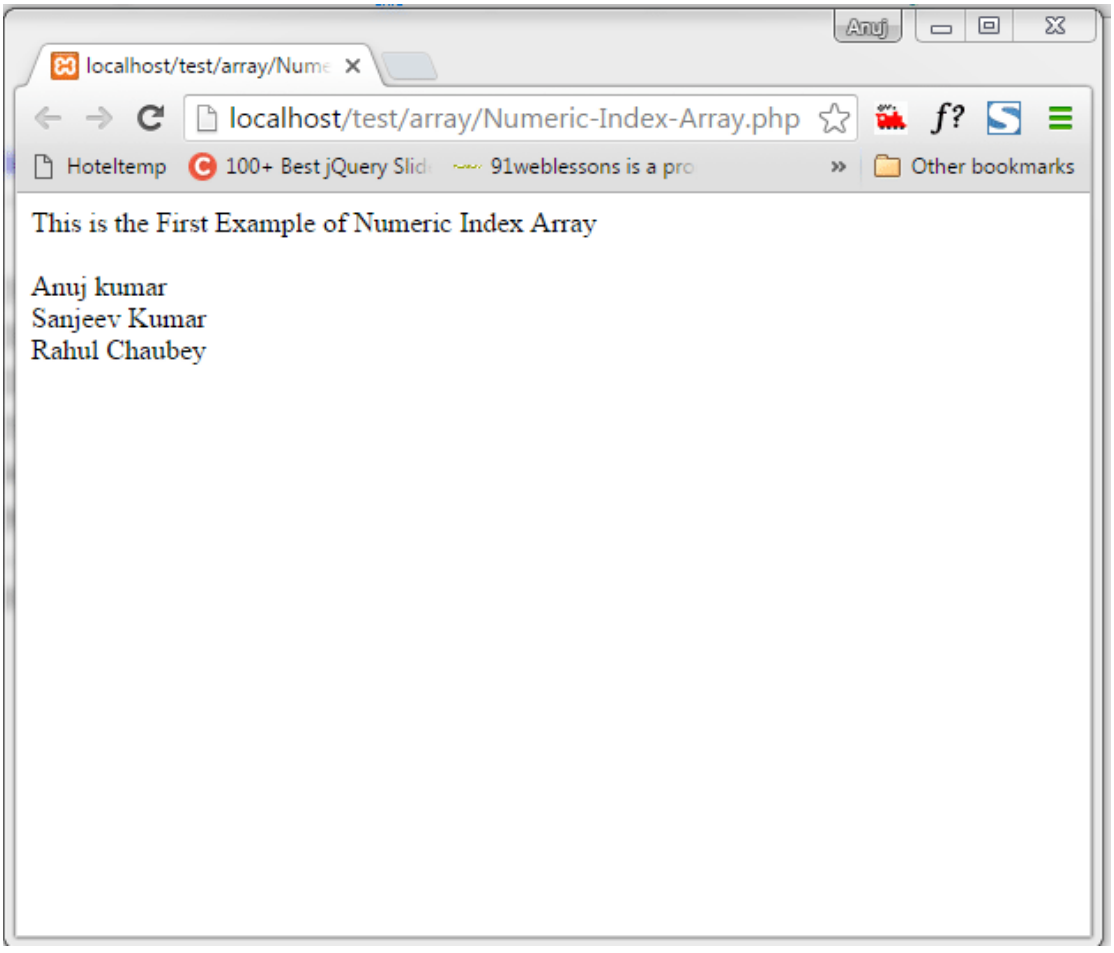
There are two ways to define the numeric index array in PHP.

Which are as follows:
1. $arrayname =array (“value1″,”value2″,”value3”);
2. $arrayname[0]=”value1″;
$arrayname[1]=”value2″;
$arrayname[2]=”value3″;

Example 1

```
<?php

$empname = array ("Anuj kumar","Sanjeev Kumar","Rahul Chaubey");

echo "This is the First Example of Numeric Index Array".'<br>'.'<br>';

echo $empname[0].'<br>';

echo $empname[1].'<br>';

echo $empname[2];

?>
```

Output



Associative Array

An Associative array also is a type of array by which you can assign an arbitrary key to every value. In an associative array, each key is associated with a value. With associative arrays we can use the values as keys and assign values to them. In an associative array, the keys are not necessarily numeric, and even when they are numeric, not necessarily in any order.

Syntax of Associative Array

There are two ways for defining the associative array in PHP. Which are as follows:

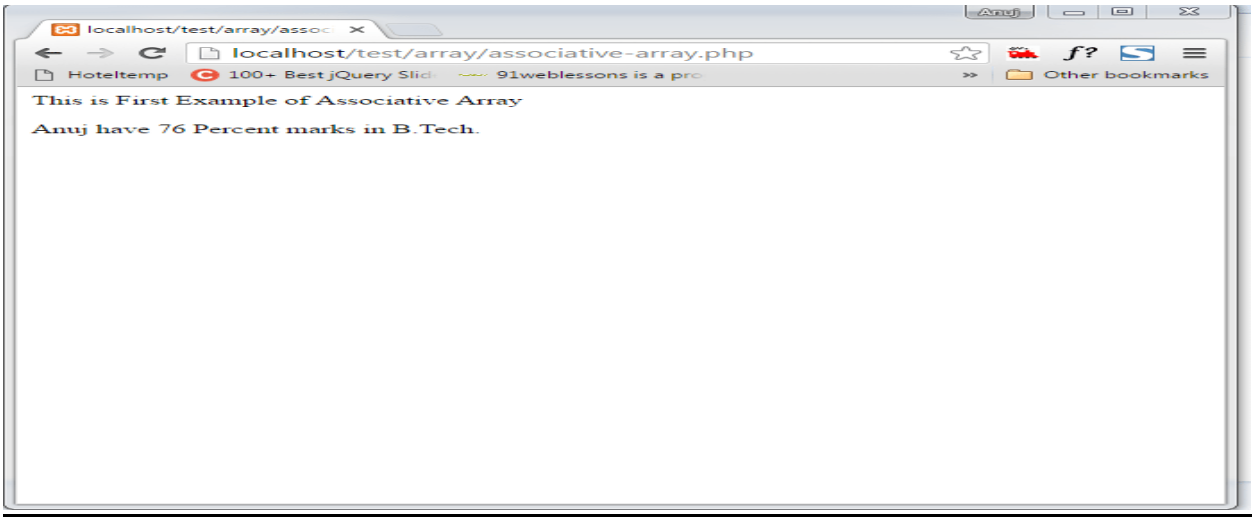
1. $arrayname = array (“value”=>20, “value2″<=30, “value3″=>40);

2. $arrayname[‘value1’] = “20”;
$arrayname[‘value2’] = “30”;
$arrayname[‘value3’] = “40”;

Example 1

```
<?php
$studentpercent ['Anuj'] = "76";
$studentpercent ['Sanjeev'] = "68";
$studentpercent ['Rahul'] = "90";
echo "This is First Example of Associative Array" .'<br>'.'<br>';
echo "Anuj have " . $studentpercent ['Anuj'] . " Percent marks in B.Tech.";
?>
```

Output



Multidimensional Array

A Multidimensional array is also known as a two-dimensional array. A Multidimensional array is an array of arrays i.e. each element in the main array can also be an array. And each element in the sub-array can be an array, and so on. A multidimensional array is an array of arrays, which means that the array can contain arrays within itself. If an array element value is another array then this is a multidimensional array.

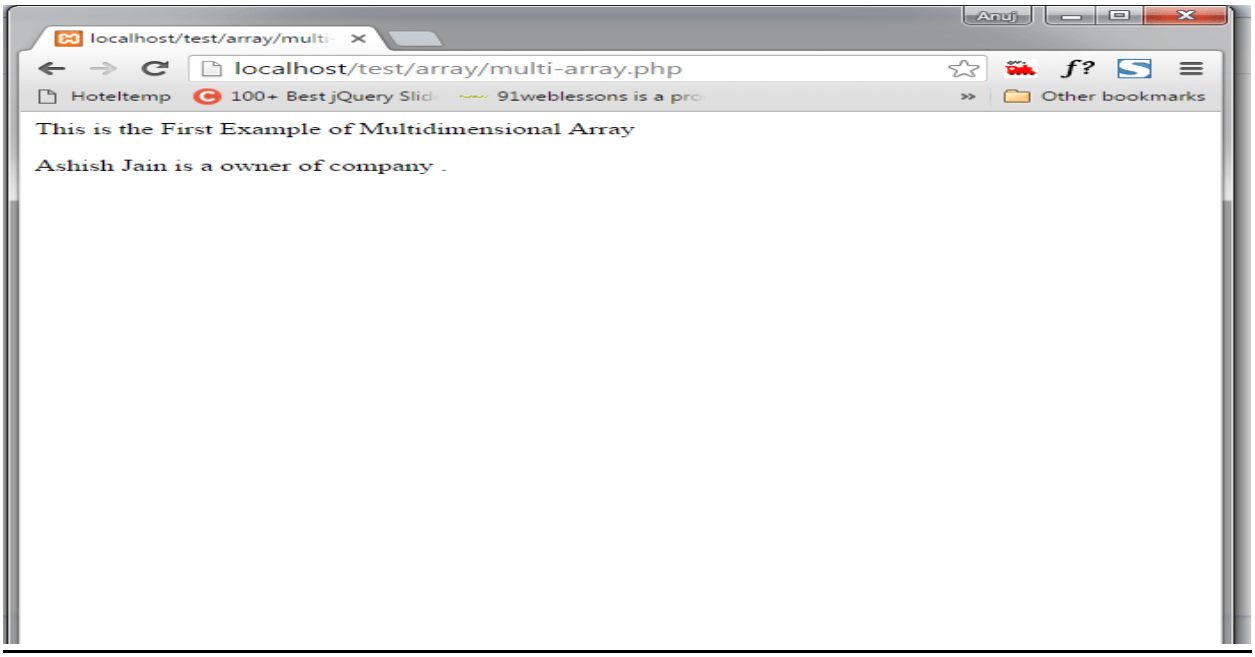
Syntax of Multidimensional Array

$arrayname = array ($array1, $array2, $array3) ;

Example1

```
<?php
$myLists['Employee'] = array("1"=>"Anuj Kumar",
 "2"=>"Sanjeev kumar",
 "3"=>"Rahul Chaubey",
 "4"=>"Atul Raguvanshi",
 "5"=>"Ashish Jain");
echo "This is the First Example of Multidimensional Array".'<br>'.'<br>';
echo $myLists ['Employee'][5] . " is a owner of company .";
?>
```

Output



4.Useful functions for arrays in PHP

sizeof($arr)

This function returns the number of elements in an array.
Use this function to find out how many elements an array contains; this information is most commonly used to initialize a loop counter when processing the array.

```
<?php

$data = array("red", "green", "blue");

echo "Array has " . sizeof($data) . " elements";

?>
```

Output

Array has 3 elements

array_values($arr)

This function accepts a PHP array and returns a new array containing only its values (not its keys). Its counterpart is the array_keys() function. Use this function to retrieve all the values from an associative array.

Ex –

```
<?php
$data = array("hero" => "Holmes", "villain" => "Moriarty");
print_r(array_values($data));
?>
```

Output

```
Array
(
[0] => Holmes
[1] => Moriarty
)
```

array_keys($arr)

This function accepts a PHP array and returns a new array containing only its keys (not its values). Its counterpart is the array_values() function.
Use this function to retrieve all the keys from an associative array.

Ex-

```
<?php
$data = array("hero" => "Holmes", "villain" => "Moriarty");
print_r(array_keys($data));
?>
```

Output:

```
Array
(
[0] => hero
[1] => villain
)
```

array_pop($arr)

This function removes an element from the end of an array.**Ex-**

```
<?php

$data = array("Donald", "Jim", "Tom");

array_pop($data);

print_r($data);

?>
```

Output

```
Array
(
[0] => Donald
[1] => Jim
)
```

array_push($arr, $val)

This function adds an element to the end of an array.

Ex-

```
<?php

$data = array("Donald", "Jim", "Tom");

array_push($data, "Harry");

print_r($data);

?>
```

Output

```
Array
(
[0] => Donald
[1] => Jim
[2] => Tom
[3] => Harry
)
```

array_shift($arr)

This function removes an element from the beginning of an array.

Ex-

```
<?php

$data = array("Donald", "Jim", "Tom");

array_shift($data);

print_r($data);

?>
```

Output

```
Array
(
[0] => Jim
[1] => Tom
)
```

array_unshift($arr, $val)

This function adds an element to the beginning of an array.

```
<?php

$data = array("Donald", "Jim", "Tom");

array_unshift($data, "Sarah");

print_r($data);

?>
```

Output:

```
Array
([0] => Sarah
[1] => Donald
[2] => Jim
[3] => Tom
)
```

each($arr)

This function is most often used to iteratively traverse an array. Each time each() is called, it returns the current key-value pair and moves the array cursor forward one element. This makes it most suitable for use in a loop.

Ex-

```
<?php

$data = array("hero" => "Holmes", "villain" => "Moriarty");

while (list($key, $value) = each($data)) {

echo "$key: $value n";

}

?>
```

Output:

hero: Holmes
villain: Moriarty

sort($arr)

This function sorts the elements of an array in ascending order. String values will be arranged in ascending alphabetical order.

Note: *Other sorting functions include asort(), arsort(), ksort(), krsort() and rsort().*

Ex-

```
<?php

$data = array("g", "t", "a", "s");

sort($data);

print_r($data);

?>
```

Output

```
Array
(
[0] => a
[1] => g
[2] => s
[3] => t
)
```

array_flip($arr)

The function exchanges the keys and values of a PHP associative array.
Use this function if you have a tabular (rows and columns) structure in an array, and you want to interchange the rows and columns.

Ex-

```
<?php

$data = array("a" => "apple", "b" => "ball");

print_r(array_flip($data));

?>
```

Output

```
Array
(
[apple] => a
[ball] => b
)
```

array_reverse($arr)

The function reverses the order of elements in an array.
Use this function to re-order a sorted list of values in reverse for easier processing—for example, when you're trying to begin with the minimum or maximum of a set of ordered values.

Ex-

```
<?php

$data = array(10, 20, 25, 60);

print_r(array_reverse($data));

?>
```

Output

```
Array
(
[0] => 60
[1] => 25
[2] => 20
[3] => 10
)
```

array_merge($arr)

This function merges two or more arrays to create a single composite array. Key collisions are resolved in favor of the latest entry.
Use this function when you need to combine data from two or more arrays into a single structure—for example, records from two different SQL queries.

Ex-

```
<?php

$data1 = array("cat", "goat");

$data2 = array("dog", "cow");

print_r(array_merge($data1, $data2));

?>
```

Output

```
Array
(
[0] => cat
[1] => goat
[2] => dog
[3] => cow
)
```

array_rand($arr)

This function selects one or more random elements from an array.
Use this function when you need to randomly select from a collection of discrete values—for example, picking a random color from a list.

Ex-

```
<?php

$data = array("white", "black", "red");

echo "Today's color is " . $data[array_rand($data)];

?>
```

Output

Today's color is red

array_search($search, $arr)

This function searches the values in an array for a match to the search term, and returns the corresponding key if found. If more than one match exists, the key of the first matching value is returned.

Use this function to scan a set of index-value pairs for matches, and return the matching index.

Ex-

```
<?php

$data = array("blue" => "#0000cc", "black" => "#000000", "green" => "#00ff00");

echo "Found " . array_search("#0000cc", $data);

?>
```

Output

```
Found blue
```

array_slice($arr, $offset, $length)

This function is useful to extract a subset of the elements of an array, as another array. Extracting begins from array offset $offset and continues until the array slice is $length elements long.

Use this function to break a larger array into smaller ones—for example, when segmenting an array by size (“chunking”) or type of data.

Ex-

```
<?php

$data = array("vanilla", "strawberry", "mango", "peaches");

print_r(array_slice($data, 1, 2));

?>
```

Output

```
Array
(
[0] => strawberry
[1] => mango
)
```

array_unique($data)

This function strips an array of duplicate values.

Use this function when you need to remove non-unique elements from an array—for example, when creating an array to hold values for a table's primary key.

Ex-

```
<?php

$data = array(1,1,4,6,7,4);

print_r(array_unique($data));

?>
```

Output

```
Array
(
[0] => 1
[3] => 6
[4] => 7
[5] => 4
)
```

array_walk($arr, $func)

This function “walks” through an array, applying a user-defined function to every element. It returns the changed array.

Use this function if you need to perform custom processing on every element of an array—for example, reducing a number series by 10%.

Ex-

```
<?php

function reduceBy10(&$val, $key) {

$val -= $val * 0.1;

}

$data = array(10,20,30,40);

array_walk($data, 'reduceBy10');

print_r($data);

?>
```

Output

```
Array
([0] => 9
[1] => 18
[2] => 27
[3] => 36
)
```

5.Sorting Arrays in PHP

Sorting is a process by which we can arrange the elements of a list in a specific order i.e. ascending or descending order. We can say sorting is the process of putting a list or a group of items in a specific order. Sorting may be alphabetical or numerical.

In PHP, sorting is a process of arranging the elements of an array in a specific order i.e. ascending or descending order. Using sorting you can analyze a list in a more effective way. You can sort an array by value, key or randomly. In the most situations we need to sort an array by value.

Some sorting functions are as follows:

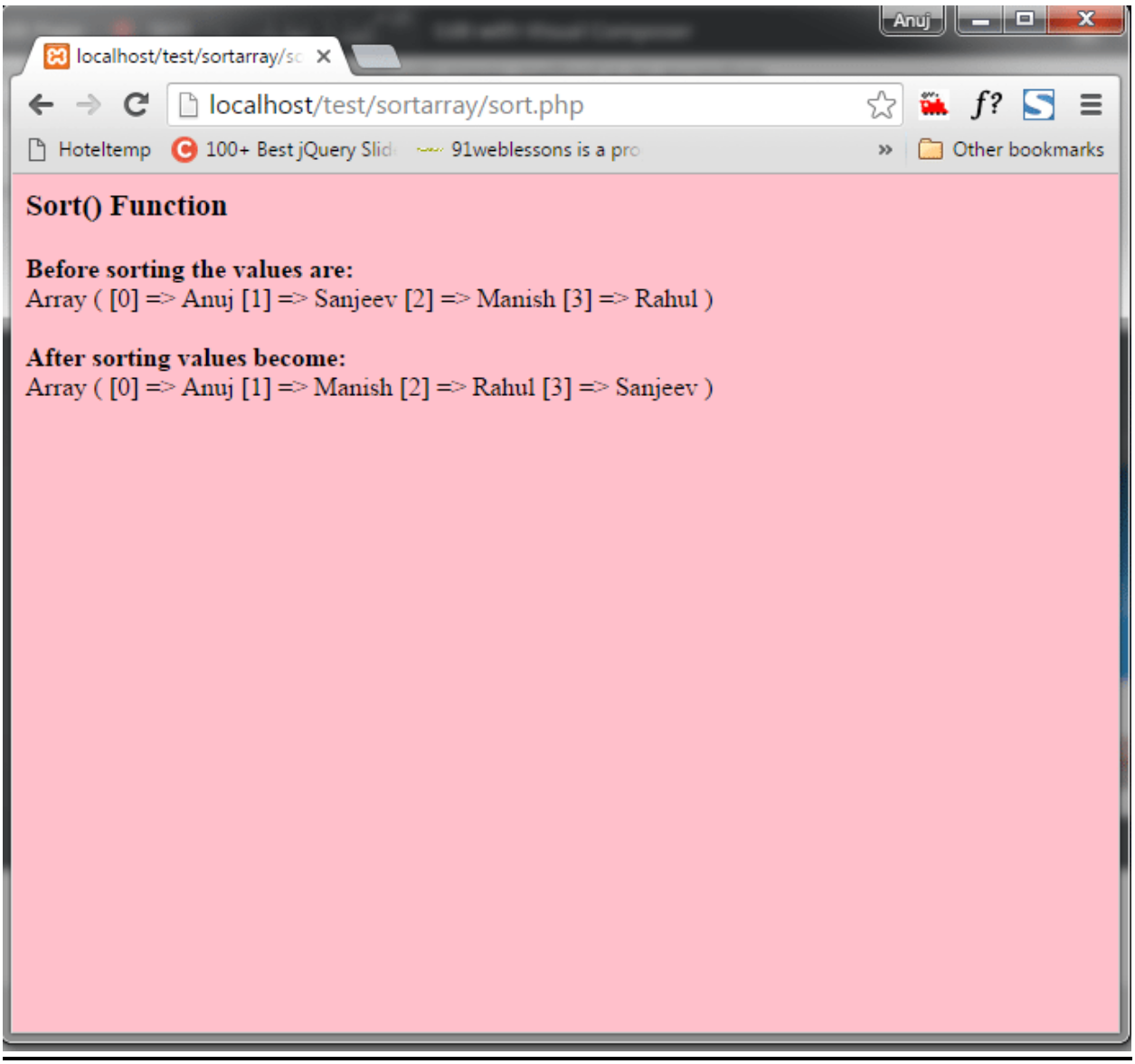
- sort()
- asort()
- rsort()
- arsort()

sort() function

Using the sort() function you can sort easily and simply. The example of the sort() function is as follows

```
<html>

<body bgcolor="pink">

<h3>Sort() Function</h3>

<?php

 $name=array ("Anuj","Sanjeev","Manish","Rahul");

 echo "<b>Before sorting the values are:</b>".'<br>';

 print_r($name).'<br>'.'<br>';

 sort($name);

 echo"<br/>";

 echo"<br/>";

 echo "<b>After sorting values become:</b>".'<br>';

 print_r($name).'<br>';

?>

</body>

</html>
```

In the above example we defined four elements (names) in the array. In this example we used the sort() function. The sort() function arranges the elements of an array in ascending (alphabetically) order. You can see in the following image


Anuj
localhost/test/sortarray/sc
localhost/test/sortarray/sort.php
Hoteltemp
100+ Best jQuery Slid
91weblessons is a pro
Other bookmarks
Sort() Function
Before sorting the values are:
Array ([0] => Anuj [1] => Sanjeev [2] => Manish [3] => Rahul)
After sorting values become:
Array ([0] => Anuj [1] => Manish [2] => Rahul [3] => Sanjeev)

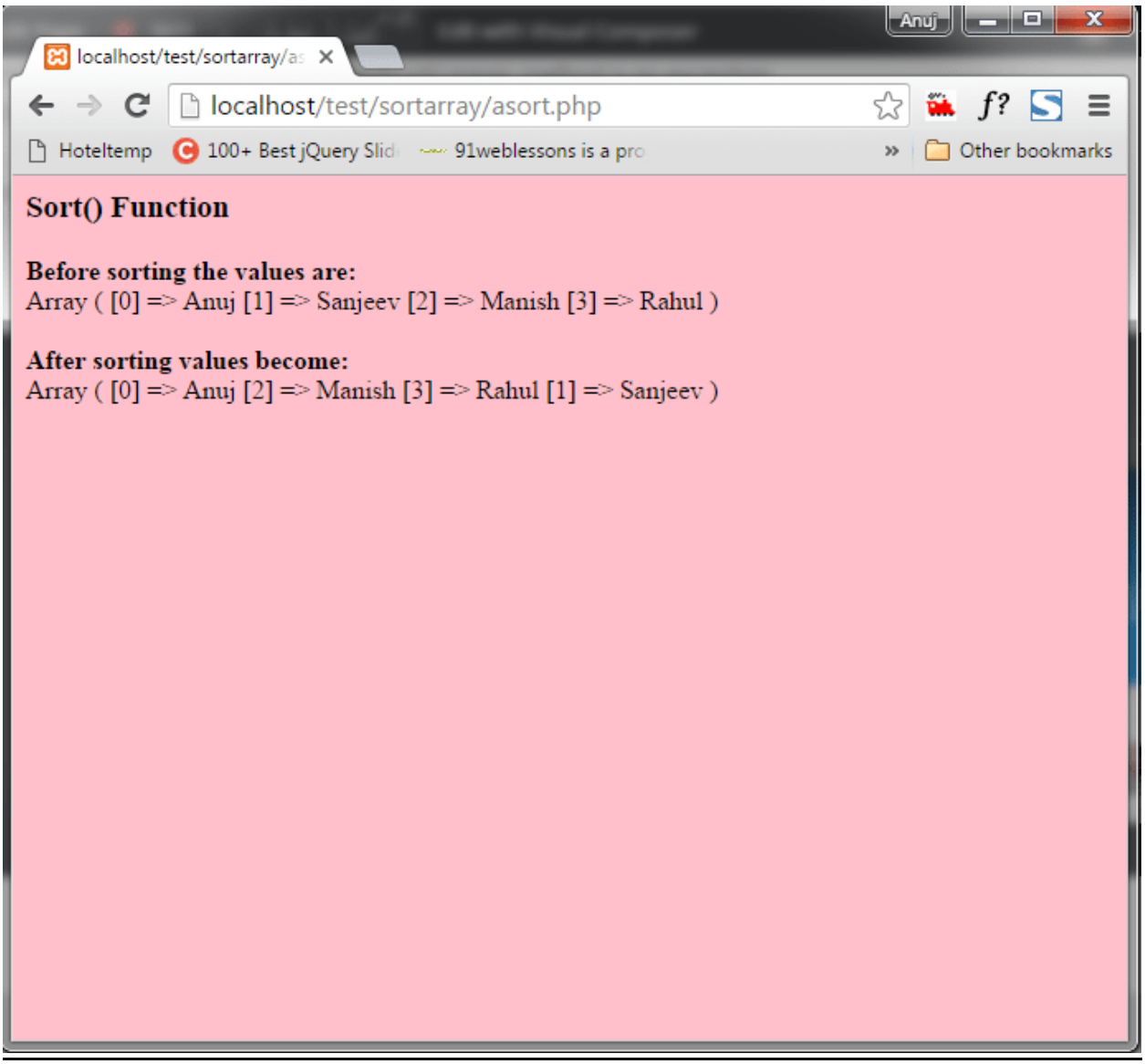
asort() function

In the example above you saw that the elements of the array are arranged in ascending (alphabetically) order. The asort() function sorts an array and also maintains the index position. An example of the asort() function is as follows:

Ex-

```
<html>
<body bgcolor="pink">
<h3>Sort() Function</h3>
<?php
 $name=array ("Anuj","Sanjeev","Manish","Rahul");
 echo "<b>Before sorting the values are:</b>".'<br>';
 print_r($name).'<br>'.'<br>';
 asort($name);
 echo"<br/>";
 echo"<br/>";
 echo "<b>After sorting values become:</b>".'<br>';
 print_r($name).'<br>';
?>
</body>
</html>
```

Output — In this output you will see the array sorted by their index position.


Anuj
localhost/test/sortarray/as
localhost/test/sortarray/asort.php
Hoteltemp
100+ Best jQuery Slid
91weblessons is a pro
Other bookmarks
Sort() Function
Before sorting the values are:
Array ([0] => Anuj [1] => Sanjeev [2] => Manish [3] => Rahul)
After sorting values become:
Array ([0] => Anuj [2] => Manish [3] => Rahul [1] => Sanjeev)

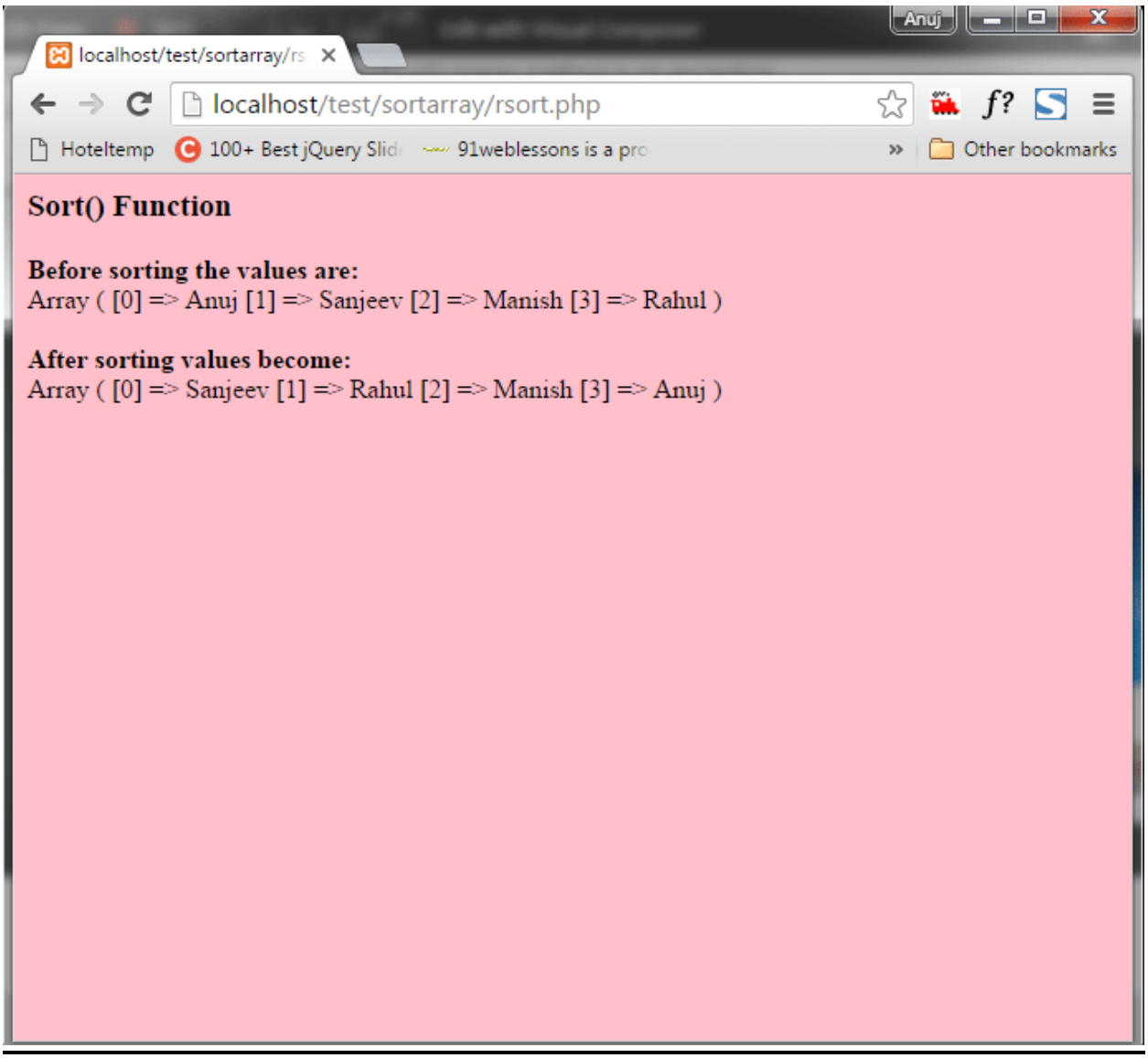
rsort() function

The rsort() function is used to sort an array in descending (alphabetic) order. An example of the rsort() function is as follows:

Ex-

```
<html>
<body bgcolor="pink">
<h3>Sort() Function</h3>
<?php
$name=array ("Anuj","Sanjeev","Manish","Rahul");
 echo "<b>Before sorting the values are:</b>".'<br>';
 print_r($name).'<br>'.'<br>';
 rsort($name);
 echo"<br/>";
 echo"<br/>";
 echo "<b>After sorting values become:</b>".'<br>';
 print_r($name).'<br>';
?>
</body>
</html>
```

Output— In this output you will see the array sorted in a descending (alphabetically) order.

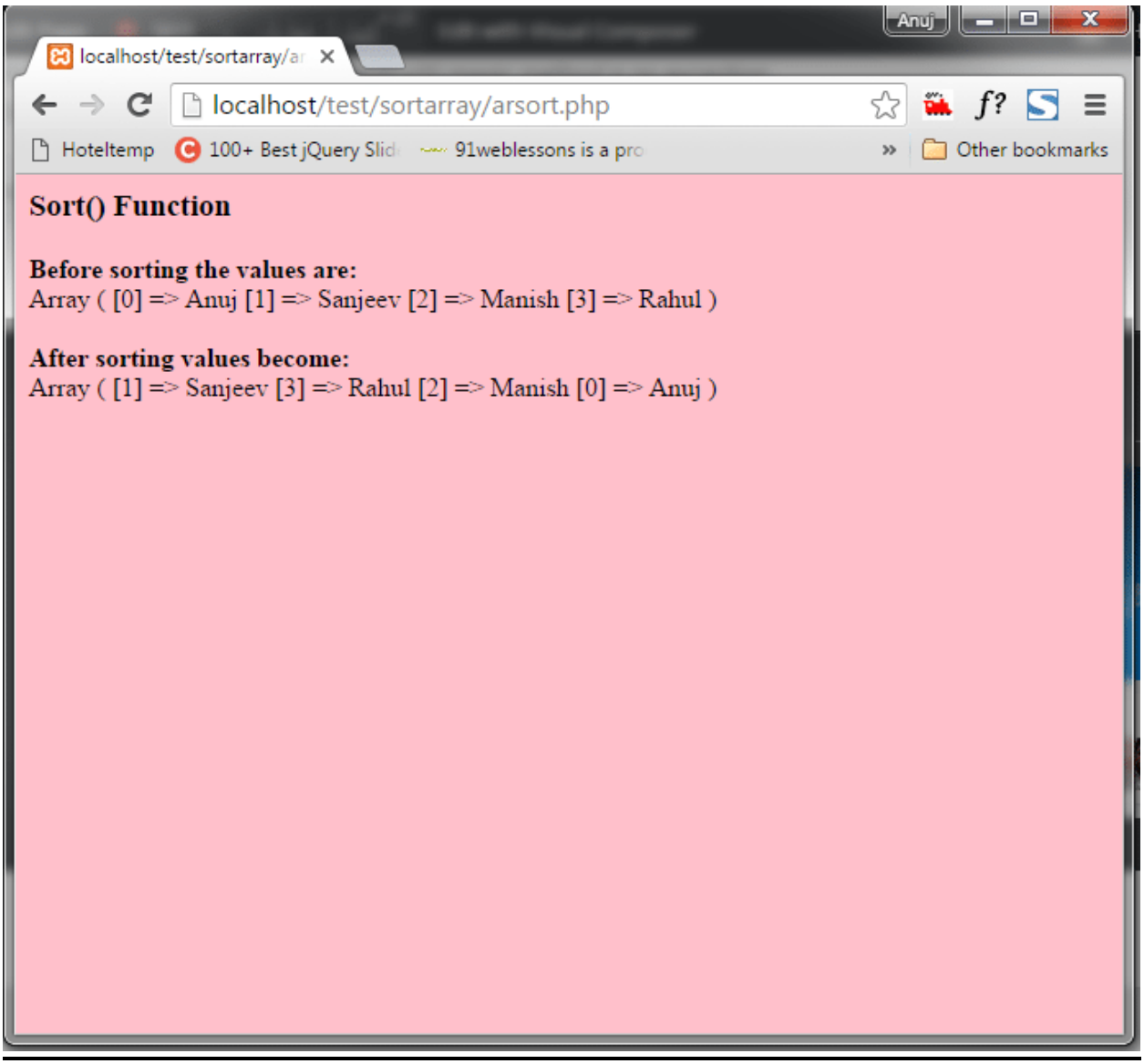

Anuj
localhost/test/sortarray/rs
localhost/test/sortarray/rsort.php
Hoteltemp
100+ Best jQuery Slid
91weblessons is a pro
Other bookmarks
Sort() Function
Before sorting the values are:
Array ([0] => Anuj [1] => Sanjeev [2] => Manish [3] => Rahul)
After sorting values become:
Array ([0] => Sanjeev [1] => Rahul [2] => Manish [3] => Anuj)

arsort() function

The arsort() function is a combination of asort() + rsort(). The arsort() function will sort an array in reverse order and maintain index position. The example of the arsort() function is as follows:

Ex-

```
<html>
<body bgcolor="pink">
<h3>Sort() Function</h3>
<?php
$name=array ("Anuj","Sanjeev","Manish","Rahul");
echo "<b>Before sorting the values are:</b>".'<br>';
print_r($name).'<br>'.'<br>';
arsort($name);
echo"<br/>";
echo"<br/>";
echo "<b>After sorting values become:</b>".'<br>';
print_r($name).'<br>';
?>
</body>
</html>
```


Anuj
localhost/test/sortarray/ar
localhost/test/sortarray/arsort.php
Hoteltemp
100+ Best jQuery Slid
91weblessons is a pro
Other bookmarks
Sort() Function
Before sorting the values are:
Array ([0] => Anuj [1] => Sanjeev [2] => Manish [3] => Rahul)
After sorting values become:
Array ([1] => Sanjeev [3] => Rahul [2] => Manish [0] => Anuj)