Working with PHP Arrays and functions

PHP Arrays

- An array variable is a storage area holding a number or text. The problem is, a variable will hold only one value.
- An array is a special variable, which can store multiple values in one single variable.

PHP Arrays

- 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="Saab";
$cars2="Volvo";
$cars3="BMW";
```

PHP Arrays

- However, what if you want to loop through the cars and find a specific one? And what if you had not 3 cars, but 300?
- The best solution here is to use an array.
- An array can hold all your variable values under a single name. And you can access the values by referring to the array name.
- Each element in the array has its own index so that it can be easily accessed.

PHP Arrays

- In PHP, there are three kind of arrays:
- **Indexed / Numeric array** - An array with a numeric index
- **Associative array** - An array where each ID key is associated with a value
- **Multidimensional array** - An array containing one or more arrays

PHP Numeric Arrays

- A numeric array stores each array element with a numeric index.
- There are two methods to create a numeric array.

PHP Numeric Arrays

- In the following example the index is automatically assigned (the index starts at 0):

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

- In the following example we assign the index manually:

```
$cars[0]="Saab";
$cars[1]="Volvo";
$cars[2]="BMW";
$cars[3]="Toyota";
```

PHP Numeric/ Indexed Arrays

- In the following example you access the variable values by referring to the array name and index:

```
<?php
$cars[0]="Saab";
$cars[1]="Volvo";
$cars[2]="BMW";
$cars[3]="Toyota";
echo $cars[0] . " and " . $cars[1] . " are Swedish cars.";
?>
```

- The code above will output:

```
Saab and Volvo are Swedish cars.
```

Building up an Array

- You can allocate a new item in the array and add a value at the same time using empty square braces [] on the right hand side of an assignment statement

```
$va = array();
$va[] = "Hello";
$va[] = "World";
print_r($va);
```

```
Array(
    [0] => Hello
    [1] => World
)
```

PHP Associative Arrays

- With an associative array, each ID key is associated with a value.
- When storing data about specific named values, a numerical array is not always the best way to do it.
- With associative arrays we can use the values as keys and assign values to them.

PHP Associative Arrays

- In this example we use an array to assign ages to the different persons:

```
$ages = array("Peter"=>32, "Quagmire"=>30, "Joe"=>34);
```

- This example is the same as the one above, but shows a different way of creating the array:

```
$ages['Peter'] = "32";
$ages['Quagmire'] = "30";
$ages['Joe'] = "34";
```

PHP Associative Arrays

The ID keys can be used in a script:

```
<?php
$ages['Peter'] = "32";
$ages['Quagmire'] = "30";
$ages['Joe'] = "34";

echo "Peter is " . $ages['Peter'] . " years old.";
?>
```

The code above will output:

```
Peter is 32 years old.
```

Looping Through an Array

```
<?php
    $stuff = array("name" => "Chuck",
                   "course" => "PHPIntro");
    foreach($stuff as $k => $v ) {
       echo "Key=",$k," Val=",$v,"\n";
    }
?>
```

```
Key=name Val=Chuck
Key=course Val=PHPIntro
```

PHP Multidimensional Arrays

- In a multidimensional array, 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.

PHP Multidimensional Arrays

In this example we create a multidimensional array, with automatically assigned ID keys:

```
$families = array
  (
  "Griffin"=>array
  (
  "Peter",
  "Lois",
  "Megan"
  ),
  "Quagmire"=>array
  (
  "Glenn"
  ),
  "Brown"=>array
  (
  "Cleveland",
  "Loretta",
  "Junior"
  )
  );
```

PHP Multidimensional Arrays

The array above would look like this if written to the output:

```
Array
(
[Griffin] => Array
   (
   [0] => Peter
   [1] => Lois
   [2] => Megan
   )
[Quagmire] => Array
   (
   [0] => Glenn
   )
[Brown] => Array
   (
   [0] => Cleveland
   [1] => Loretta
   [2] => Junior
   )
)
```

PHP Multidimensional Arrays

Lets try displaying a single value from the array above:

```
echo "Is " . $families['Griffin'][2] .
" a part of the Griffin family?";
```

The code above will output:

```
Is Megan a part of the Griffin family?
```

Arrays of Arrays

The elements of an array can be many things other than a string or integer. You can even have objects or other arrays.

```
$products = array(
    'paper' =>      array(
        'copier' => "Copier & Multipurpose",
        'inkjet' => "Inkjet Printer",
        'laser' => "Laser Printer",
        'photo' => "Photographic Paper"),
    'pens' => array(
        'ball'      => "Ball Point",
        'hilite' => "Highlighters",
        'marker' => "Markers"),
    'misc' => array(
        'tape'      => "Sticky Tape",
        'glue'      => "Adhesives",
        'clips' => "Paperclips")
);
```

```
echo $products["pens"]["marker"];

Markers
```

Arrays and Strings

```
$inp = "This is a sentence with seven
words";$temp = explode(' ', $inp);
print_r($temp);
```

```
Array(
    [0] => This
    [1] => is
    [2] => a
    [3] => sentence
    [4] => with
    [5] => seven
    [6] => words
)
```

Array Functions

- array_push()
- array_pop()
- array_key_exists()
- array_merge()
- Print_r()
- count()
- is_array()
- isset()
- sort()
- ksort()
- asort()
- shuffle()
- Array_flip()
- array_change_key_case ()
- array_search ()
- array_sum ()
- For more function can visit here...https://www.w3schools.com/php/php_ref_array.asp

PHP array_push() Function

- The array_push() function inserts one or more elements to the end of an array.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.

Syntax

array_push(*array,value1,value2...*)

Parameter : Array as input where we will add elements and value which want to add in array on element minimum required.

Return Value : Returns the new number of elements in the array

PHP array_push() Function

PHP Script :

```
<!DOCTYPE html>
<html>
<body>
<?php
$a=array("DWPD","CM","JAVA");
array_push($a,"PROJECT-I","PROJECT-II");
print_r($a);
?>
</body>
</html>
```

OutPut :

Array ([0] => DWPD [1] => CM [2] => JAVA [3] => PROJECT-I [4] => PROJECT-II)

PHP array_pop() Function

- The array_pop() function deletes the last element of an array.

Syntax

array_pop(*array*)

Parameter : Array to remove last element mandatory.

Return Value : Returns the last value of *array*. If *array* is empty, or is not an array, NULL will be returned.

PHP array_pop() Function

PHP Script :

```
<!DOCTYPE html>
<html><body>
<?php
$a=array("DWPD","CM","JAVA");
array_push($a,"PROJECT-I","PROJECT-II");
echo "Array after Push Operation ,</br>";
print_r($a); echo "</br>";
echo "Array after Pop Operation ,</br>";
Array_pop($a);  print_r($a);echo "</br>";
?>
</body>
</html>
```

Output :

```
Array after Push Operation ,
Array ( [0] => DWPD [1] => CM [2] => JAVA [3] => PROJECT-I [4] => PROJECT-II )
Array after Pop Operation ,
Array ( [0] => DWPD [1] => CM [2] => JAVA [3] => PROJECT-I )
```

PHP array_key_exists() Function

- 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.

Syntax

array_key_exists(*key,array*)

Parameter : two parameters first array where we want to check key and second key which we want to find in array.

Return Value : Returns TRUE if the key exists and FALSE if the key does not exist.

PHP array_key_exists() Function

PHP Script :

```
<?php
$a=array("DWPD"=>"APACHE","JAVA"=>"TOMCAT");
if (array_key_exists("DWPD",$a))
  {
  echo "Key exists!";
  }
else
  {
  echo "Key does not exist!";
  }
?>
```

Output :

Key exists!

PHP array_merge() Function

- The array_merge() function merges one or more arrays into one array.You can assign one array to the function, or as many as you like.
- If two or more array elements have the same key, the last one overrides the others.If you assign only one array to the array_merge() function, and the keys are integers, the function returns a new array with integer keys starting at 0 and increases by 1 for each value.

Syntax

array_merge(*array1,array2,array3...*)

Parameter : Arrays want to merge minimum one array is mandatory to pass as Input.

Return Value : Returns merged array.

PHP array_merge() Function

PHP Script :

```
<?php
$a1=array("DWPD","JAVA");
$a2=array("PROJ-I","PROJ-II");
$a3["3RDSEM"]="C++";
$a3[1]="MALP";
$a3['4THSEM']=".NET";
print_r(array_merge($a1,$a2,$a3));
?>
```

Output :

Array ([0] => DWPD [1] => JAVA [2] => PROJ-I [3] => PROJ-II [3RDSEM] => C++ [4] => MALP [4THSEM] => .NET)

Print_r()

```
<?php
    $arr = array(1,2,3,4,5);
    Print_r($arr);
?>
```

```
Array ( [0] => 1 [1] => 2 [2] => 3 [3] => 4 [4] => 5 )
```

PHP Count() Function

- This is build in function ,counts all elements in an array, or something in an object.
  Syntax

  count($array , mode)

Parameter : 1. Array to count the elements of array.

- 2. Mode 0 for one dimensional array optional.
- **Note:** If you want to count one dimensional array then option set 0 but default its zero so there is no requirement to set zero.In case of multidimensional array mode will set 1, it's compulsory.
- **Return Value** : It returns number of element in an array.

Count Function - Example

```
<?php
  $a[0] = 1;
  $a[1] = 3;
  $a[2] = 5;
  $result = count($a); // $result == 3
  $b[0]  = 7;
  $b[5]  = 9;
  $b[10] = 11;
  $result = count($b); // $result == 3
  $result = count(null);// $result == 0
  $result = count(false);// $result == 1
  ?>
```

Count Function - Example

```
<?php
        $array = array("M", "u","n","e","e","r");
        for ($i = 0; $i < count($array); $i++)
        {
                echo  $array[$i];
        }
?>
```

PHP : is_array() function

- The is_array() function is used to find whether a variable is an array or not.

Syntax

is_array (var_name);

Parameter : variable that needs to check for array and it is mandatory.

Return Value : TRUE if var is an array, FALSE otherwise.

PHP : is_array() function

- PHP Script :

```
<?php
$var_name=array('A','B','C');
if (is_array($var_name))
echo 'This is an array....';
else
echo 'This is not an array....';
?>
```

Output :

This is an array....

PHP current() function

- 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.
- This function does not move the arrays internal pointer.

Syntax

current(*array*) ;

Parameter : Array to find the current position.

Return Value : Returns the value of the current element in an array, or FALSE on empty elements or elements with no value.

- Related methods:
  - ✓ end() - moves the internal pointer to, and outputs, the last element in the array
  - ✓ next() - moves the internal pointer to, and outputs, the next element in the array
  - ✓ prev() - moves the internal pointer to, and outputs, the previous element in the array
  - ✓ reset() - moves the internal pointer to the first element of the array
  - ✓ each() - returns the current element key and value, and moves the internal pointer forward.

PHP current() function

- PHP Script :

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


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

Output :

Peter

PHP in_array() Function

- The in_array() function searches an array for a specific value.
- If the search parameter is a string and the type parameter is set to TRUE, the search is case-sensitive.

Syntax

in_array(search,array,type)

Parameter :

1. **Search** : Specifies the what to search for search.
2. **Array :** Array that needs to be search
3. **Type :** If this parameter is set to TRUE, the in_array() function searches for the search-string and specific type in the array.

Return Value :Returns TRUE if the value is found in the array, or FALSE otherwise

PHP in_array() Function

PHP Script :

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

 if (in_array("Glenn", $people))
 {
          echo "Match found";
 }
else
 {
         echo "Match not found";
 }
?>
```

Output :

Match found

PHP Array Search Function

- array_search() function is used to find the element of an array and return its key. And after that you can access the element with help of key.

Syntax

```
array_search(value,array,strict) ;
```

Parameter :

1. **Value :** String that needs to search in array.
2. **Array :** array to search in side array.
3. **Strict :** If this parameter is set to TRUE, then this function will search for identical elements in the array. Possible values are true and false if set true then Number 5 and String 5 are different.by default it remains false optional parameter.

Return Value : Returns the key of a value if it is found in the array, and FALSE otherwise. If the value is found in the array more than once, the first matching key is returned.

PHP Array Search Function

- PHP Script :

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

Output :

a

PHP : isset() function

- The isset () function is used to check whether a variable is set or not. If a variable is already unset with unset() function, it will no longer be set. The isset() function return false if testing variable contains a NULL value.

Syntax :

isset(variable1, variable2.....)

Parameter : Variable as parameter needs to be checked is set or not.

Return value : TRUE if variable (variable1,variable2..) exists and has value not equal to NULL, FALSE otherwise

PHP : isset() function

PHP Script :

```
<?php
$var1 = 'test';
var_dump(isset($var1));
?>
```

Output :

bool(true)

PHP sort() Function

- The sort() function sorts an indexed array in ascending order.
- Use the rsort() function to sort an indexed array in descending order.

Syntax

sort(*array,sortingtype*);

Parameter : 1. Array that needs to sort mandatory.

2. Sortingtype : how it will compare

- 0 = SORT_REGULAR - Default. Compare items normally (don't change types)
- 1 = SORT_NUMERIC - Compare items numerically
- 2 = SORT_STRING - Compare items as strings
- 3 = SORT_LOCALE_STRING - Compare items as strings, based on current locale
- 4 = SORT_NATURAL - Compare items as strings using natural ordering.

Return Value : TRUE on success. FALSE on failure.

PHP sort() Function

- **PHP Script :**

```
<?php
$cars=array(“DWPD",“JAVA”,“CM“,”PROJECT-I”);
sort($cars);

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

Output :

CM
DWPD
JAVA
PROJECT-I

PHP krsort() Function

- The krsort() function sorts an associative array in descending order, according to the key.

Syntax

krsort(*array,sortingtype*);

Parameter : 1. Array that needs to sort mandatory.

2. Sortingtype : how it will compare

- 0 = SORT_REGULAR - Default. Compare items normally (don't change types)
- 1 = SORT_NUMERIC - Compare items numerically
- 2 = SORT_STRING - Compare items as strings
- 3 = SORT_LOCALE_STRING - Compare items as strings, based on current locale
- 4 = SORT_NATURAL - Compare items as strings using natural ordering.

Return Value : TRUE on success. FALSE on failure.

PHP krsort() Function

- **PHP Script :**

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

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

Output :

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

PHP arsort() Function

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

Syntax

arsort(*array,sortingtype*);

Parameter : 1. Array that needs to sort mandatory.

2. Sortingtype : how it will compare

- 0 = SORT_REGULAR - Default. Compare items normally (don't change types)
- 1 = SORT_NUMERIC - Compare items numerically
- 2 = SORT_STRING - Compare items as strings
- 3 = SORT_LOCALE_STRING - Compare items as strings, based on current locale
- 4 = SORT_NATURAL - Compare items as strings using natural ordering.

Return Value : TRUE on success. FALSE on failure.

PHP arsort() Function

- PHP Script :

```
<?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>";
  }
?>
```

Output :

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

PHP shuffle() Function

- The shuffle() function randomizes the order of the elements in the array.
- This function assigns new keys for the elements in the array. Existing keys will be removed.

Syntax

shuffle(*array*)

Parameter : Array that needs to shuffle.

Return Value: Returns TRUE on success or FALSE on failure.

PHP shuffle() Function

PHP Script :

```
<?php
$my_array =
array("red","green","blue","yellow","purple");

shuffle($my_array);
print_r($my_array);
?>
```

Output :

Array ([0] => blue [1] => purple [2] => green [3] => yellow [4] => red)

PHP Array Change Key Case Function

This function is used to change the case of arrays key.

Syntax

```
array_change_key_case($array, option);
```

Parameter : Array Any array variable (required),

Option Default: 0 (lowercase) 1:(uppercase)

Default option value is zero.

Return Value : It returns an array keys lowercase or uppercase as per condition.

PHP Array Change Key Case Function

PHP Script :

```
<?php
$student = array
(
"NamE" => "JOHN",
"RolL NumbeR" => 38
);
print_r(array_change_key_case($student,1));
?>
```

Output Of Given PHP Script

Array ([NAME] => JOHN [ROLL NUMBER] => 38)

PHP array_sum() Function

The array_sum() function returns the sum of all the values in the array.

Syntax

array_sum(*array*);

Parameter : array as input that needs to sum all elements.

Return Value : Returns the sum of all the values in an array

PHP array_sum() Function

- **PHP Script :**

```
<?php
$a=array(5,34,46,15,25);
echo array_sum($a);
?>
```

Output :

125

PHP array_flip() Function

- array_flip function is used to exchange the key as value and value as a key.
- **Note:** Keep in mind that array_flip() only flips the key/value mapping and does not reverse the positioning.

Syntax

```
array_flip ( $array );
```

Parameter : array as input to flip.

Return Value :It returns an array with the keys of $array as values, and the values of $array as the new keys.

PHP array_flip() Function

Php Script :

```
<?php
$array = array
(
   "a" => "apple",  "b" => "banana",  "o"=>"orange"
);
print_r( array_flip($array) );
?>
```

Output Of Given Php Script

Array ([apple] => a [banana] => b [orange] => o)

PHP Functions

- We will now explore how to create your own functions.
- To keep the script from being executed when the page loads, you can put it into a function.
- A function will be executed by a call to the function.
- You may call a function from anywhere within a page.

PHP Functions

- A function will be executed by a call to the function.

```
function functionName()
{
code to be executed;
}
```

- Give the function a name that reflects what the function does
- The function name can start with a letter or underscore (not a number)

PHP Functions

- A simple function that writes a name when it is called:

```
<html>
<body>

<?php
function writeName()
{
echo "Kai Jim Refsnes";
}

echo "My name is ";
writeName();
?>

</body>
</html>
```

PHP Functions - Parameters

- Adding parameters...
- To add more functionality to a function, we can add parameters. A parameter is just like a variable.
- Parameters are specified after the function name, inside the parentheses.

PHP Functions - Parameters

The following example will write different first names, but equal last name:

```
<html>
<body>

<?php
function writeName($fname)
{
echo $fname . " Refsnes.<br />";
}

echo "My name is ";
writeName("Kai Jim");
echo "My sister's name is ";
writeName("Hege");
echo "My brother's name is ";
writeName("Stale");
?>

</body>
</html>
```

PHP Functions - Parameters

Output:

```
My name is Kai Jim Refsnes.
My sister's name is Hege Refsnes.
My brother's name is Stale Refsnes.
```

PHP Functions - Parameters

```
<html>
<body>

<?php
function writeName($fname,$punctuation)
{
echo $fname . " Refsnes" . $punctuation . "<br />";
}

echo "My name is ";
writeName("Kai Jim",".");
echo "My sister's name is ";
writeName("Hege","!");
echo "My brother's name is ";
writeName("Ståle","?");
?>

</body>
</html>
```

This example adds different punctuation.

PHP Functions - Parameters

Output:

```
My name is Kai Jim Refsnes.
My sister's name is Hege Refsnes!
My brother's name is Ståle Refsnes?
```

PHP Function with Parameter

PHP Script :

```
<?php
    function addFunction($num1, $num2) {
      $sum = $num1 + $num2;
      echo "Sum of the two numbers is : $sum";
    }
    addFunction(40, 20);
  ?>
```

Output :

Sum of the two numbers is : 60

Passing Arguments by Reference

- It is possible to pass arguments to functions by reference. This means that a reference to the variable is manipulated by the function rather than a copy of the variable's value.
- Any changes made to an argument in these cases will change the value of the original variable. You can pass an argument by reference by adding an ampersand to the variable name in either the function call or the function definition.

Passing Arguments by Reference

PHP Script :

```
<?php
    function addFive($num) {
      $num += 5;
    }
    function addSix(&$num) {
      $num += 6;
    }
    $orignum = 10;
    addFive( $orignum );
    echo "Original Value is $orignum<br />";
          addSix( $orignum );
    echo "Original Value is $orignum<br />";
  ?>
```

Output :

Original Value is 10

Original Value is 16

PHP Functions returning value

- A function can return a value using the **return** statement in conjunction with a value or object. return stops the execution of the function and sends the value back to the calling code.
- You can return more than one value from a function using **return array(1,2,3,4)**.
- Following example takes two integer parameters and add them together and then returns their sum to the calling program. Note that **return** keyword is used to return a value from a function.

PHP Functions returning value

PHP Script :

```
<?php
    function addFunction($num1, $num2) {
      $sum = $num1 + $num2;
      return $sum;
    }
    $return_value = addFunction(10, 20);

    echo "Returned value from the function : $return_value";
  ?>
```

Output :

Returned value from the function : 30

Setting Default Values for Function Parameters

- You can set a parameter to have a default value if the function's caller doesn't pass it.
- Following function prints NULL in case use does not pass any value to this function.

Setting Default Values for Function Parameters

PHP Script :

```
<?php
    function printMe($param = NULL) {
      print $param;
    }

    printMe("This is test");
    printMe();
  ?>
```

Output :

This is test

Dynamic Function Calls

- It is possible to assign function names as strings to variables and then treat these variables exactly as you would the function name itself. Following example depicts this behaviour.
- PHP Script :

```
<?php
    function sayHello() {
      echo "Hello<br />";
    }

    $function_holder = "sayHello";
    $function_holder();
  ?>
```

Output :

Hello

Variable length Argument list

- Gets an array of the function's argument list.

Syntax

array func_get_args ( )

- Gets an array of the function's argument list.
- This function may be used in conjunction with func_get_arg() and func_num_args() to allow user-defined functions to accept variable-length argument lists.

Return Values : Returns an array in which each element is a copy of the corresponding member of the current user-defined function's argument list.

Variable length Argument list

- **PHP Script :**

```
function Average()
{
   $result = 0;
   $arguments = func_get_args();
   foreach ($arguments as $argument)
   {
      $result += $argument;
   }
   return ($result / max(1, func_num_args()));
}
echo Average(1, 2, 3, 4, 5); // 3
```

Output :

3

PHP MATH FUNCTIONS

- PHP Abs() Function
- PHP Ceil() Function
- PHP Floor() Function
- PHP Round() Function
- PHP Min() Function
- PHP Max() Function
- PHP Pow() Function
- PHP Sqrt() Function
- PHP Mt_rand() Function
- PHP Frmod() Function
- PHP rand() Function
- https://www.w3schools.com/php/php_ref_math.asp

PHP Abs() Function

Abs() function used to get absolute values of the given parameter. We can directly pass the integer or float into the abs function or store on a variable,pass the variable into the abs function.

Syntax

abs(number);

Parameter : Number integer or float value.

Return Value :The absolute value of number. If the argument number is of type float, the return type is also float, otherwise it is integer (as float usually has a bigger value range than integer).

PHP Abs() Function

PHP Script :

```
<?php
echo abs(-38);
var_dump( abs(-38) );
?>
```

Output :

```
38
int 38
```

PHP Ceil() Function

- The ceil() function rounds a number UP to the nearest integer, if necessary.
- **Tip:** To round a number DOWN to the nearest integer, look at the floor() function.
- **Tip:** To round a floating-point number, look at the round() function.

Syntax

```
ceil(number);
```

Parameter : Number integer or float value.

Return Value : rounds a number UP to the nearest integer.

PHP Ceil() Function

PHP Script :

```
<!DOCTYPE html>
<html>
<body>
<?php
echo(ceil(0.60) . "<br>");
echo(ceil(0.40) . "<br>");
echo(ceil(5) . "<br>");
echo(ceil(5.1) . "<br>");
echo(ceil(-5.1) . "<br>");
echo(ceil(-5.9));
?>
</body>
</html>
```

Output :

```
1
1
5
6
-5
-5
```

PHP Floor() Function

- The floor() function rounds a number DOWN to the nearest integer, if necessary.

Syntax

floor(*number*);

Parameter : Number integer or float value.

Return Value : The value rounded down to the nearest integer

PHP Floor() Function

PHP Script :

```
<!DOCTYPE html>
<html>
<body>

<?php
echo(floor(0.60) . "<br>");
echo(floor(0.40) . "<br>");
echo(floor(5) . "<br>");
echo(floor(5.1) . "<br>");
echo(floor(-5.1) . "<br>");
echo(floor(-5.9));
?>

</body>
</html>
```

Output :

```
0
0
5
5
-6
-6
```

PHP Round() Function

- The round() function rounds a floating-point number.

Syntax

```
round(number,precision,mode);
```

Parameter :

- **Number** : Required. Specifies the value to round.
- **Precision** : Optional. Specifies the number of decimal digits to round to. Default is 0
- **Mode** : Optional. Specifies a constant to specify the rounding mode.

PHP Round() Function

Rounding Modes

•PHP_ROUND_HALF_UP - Default. Rounds number up to precision decimal, when it is half way there. Rounds 1.5 to 2 and -1.5 to -2

•PHP_ROUND_HALF_DOWN - Round number down to precision decimal places, when it is half way there. Rounds 1.5 to 1 and -1.5 to -1

•PHP_ROUND_HALF_EVEN - Round number to precision decimal places towards the next even value

•PHP_ROUND_HALF_ODD - Round number to precision decimal places towards the next odd value

PHP Round() Function

PHP Script :

```
<!DOCTYPE html>
<html>
<body>

<?php
echo(round(0.60) . "<br>");
echo(round(0.50) . "<br>");
echo(round(0.49) . "<br>");
echo(round(-4.40) . "<br>");
echo(round(-4.60));
?>

</body>
</html>
```

Output :

```
1
1
0
-4
-5
```

PHP Min() and Max() Function

- The max() function is used to calculate the maximum number of various numbers. The min() function is used to calculate minimum number of various numbers. The example of the max() and min() functions is as follows:

Syntax

min(*array_values*);

or

min(*value1,value2,...*);

max(*array_values*);

or

max(*value1,value2,...*);

Parameter : Numeric Value and Array

Return Value : Returns minimum or maximum number from given Number or Array.

PHP Min() and Max() Function

PHP Script :

```
<!DOCTYPE html>
<html>
<body>
<?php
echo(max(2,4,6,8,10) . "<br>");
echo(max(array(44,16,81,12)));
echo(min(22,14,68,18,15) . "<br>");
echo(min(array(4,6,8,10)) . "<br>");
?>
</body>
</html>
```

Output :

```
10
81
14
4
```

PHP Pow() Function

- The pow() function returns x raised to the power of y.

Syntax

```
pow(x,y);
```

Parameters : *x* Mandatory Number for base value , Y Number value specifies the exponent.

Return Value: x raised to the power of y

PHP Pow() Function

PHP Script :

```
<!DOCTYPE html>
<html>
<body>

<?php
echo(pow(2,4) . "<br>");
echo(pow(-2,4) . "<br>");
echo(pow(-2,-4) . "<br>");
echo(pow(-2,-3.2));
?>

</body>
</html>
```

Output :

16
16
0.0625
NAN

PHP sqrt() Function

- The sqrt() function returns the square root of a number.

Syntax

sqrt(*number*);

Parameter : Number to get Square root of number.

Return Value : Square root of input number.

PHP sqrt() Function

PHP Script :

```
<!DOCTYPE html>
  <html>
  <body>

  <?php
  echo(sqrt(0) . "<br>");
  echo(sqrt(1) . "<br>");
  echo(sqrt(9) . "<br>");
  echo(sqrt(0.64) . "<br>");
  echo(sqrt(-9));
  ?>

  </body>
  </html>
```

Output :

```
0
1
3
0.8
NAN
```

PHP Mt_rand() Function

- Mt_rand function is used to find the random number in between min and max.
- It Returns the better random number than rand function.

Syntax

```
mt_rand([min, max]);
```

Parameters : Min & Max two parameters Integer or float values. Both parameters are optional.

Return Value : It returns random number from given range of min, max numbers.

PHP Mt_rand() Function

Php Script :

```
<?php
echo mt_rand();
?>
```

Output :

2324

PHP fmod() Function

- Fmod() function returns the remainder (modulo) of the arguments.

Syntax

Fmod(float number1,float number2)

Parameters : two numbers for division.

Return Value :remainder (modulo) of the arguments.

PHP fmod() Function

- **PHP Script**

```
<?php
$x = 7;
$y = 2;
$result = fmod($x,$y);
echo $result;
// $result equals 1, because 2 * 3 + 1 = 7
?>
```

OUTPUT

1

PHP Date Functions

- Date()
- Getdate()
- CheckDate()
- Time()
- Mktime()

PHP date() Function

- The date() function formats a local date and time, and returns the formatted date string.

Syntax

date(format,timestamp);

Parameter :1. **Format** Specifies the format of the outputted date string and it is mandatory.

2.Timestamp : Specifies an integer Unix timestamp. Default is the current local time (time()) and Optional.

Date() Function format

- d - The day of the month (from 01 to 31)
- D - A textual representation of a day (three letters)
- j - The day of the month without leading zeros (1 to 31)
- S - The English ordinal suffix for the day of the month (2 characters st, nd, rd or th. Works well with j)
- F - A full textual representation of a month (January through December)
- l (lowercase 'L') - A full textual representation of a day
- m - A numeric representation of a month (from 01 to 12)
- M - A short textual representation of a month (three letters)
- n - A numeric representation of a month, without leading zeros (1 to 12)
- Y - A four digit representation of a year
- y - A two digit representation of a year
- a - Lowercase am or pm
- A - Uppercase AM or PM
- h - 12-hour format of an hour (01 to 12)
- H - 24-hour format of an hour (00 to 23)
- i - Minutes with leading zeros (00 to 59)

Date() Function format

```
<!DOCTYPE html>
<html>
<body>
<?php
// Prints the day
echo date("l") . "<br>";
// Prints the day, date, month, year, time, AM or PM
echo date("l jS \of F Y h:i:s A") . "<br>";
// Prints October 3, 1975 was on a Friday
echo "Oct 3,1975 was on a ".date("l", mktime(0,0,0,10,3,1975)) . "<br>";
// Use a constant in the format parameter
echo date(DATE_RFC822) . "<br>";
?>
</body>
</html>
```

Thursday
Thursday 24th of August 2017 03:09:42 AM
Oct 3,1975 was on a Friday
Thu, 24 Aug 17 03:09:42 -0400
1975-10-03T00:00:00-04:00

Date() Function format

- DATE_ATOM - Atom (example: 2013-04-12T15:52:01+00:00)
- DATE_COOKIE - HTTP Cookies (example: Friday, 12-Apr-13 15:52:01 UTC)
- DATE_ISO8601 - ISO-8601 (example: 2013-04-12T15:52:01+0000)
- DATE_RFC822 - RFC 822 (example: Fri, 12 Apr 13 15:52:01 +0000)
- DATE_RFC850 - RFC 850 (example: Friday, 12-Apr-13 15:52:01 UTC)
- DATE_RFC1036 - RFC 1036 (example: Fri, 12 Apr 13 15:52:01 +0000)
- DATE_RFC1123 - RFC 1123 (example: Fri, 12 Apr 2013 15:52:01 +0000)
- DATE_RFC2822 - RFC 2822 (Fri, 12 Apr 2013 15:52:01 +0000)
- DATE_RFC3339 - Same as DATE_ATOM (since PHP 5.1.3)
- DATE_RSS - RSS (Fri, 12 Aug 2013 15:52:01 +0000)
- DATE_W3C - World Wide Web Consortium (example: 2013-04-12T15:52:01+00:00)

PHP getdate() Function

- The getdate() function returns date/time information of a timestamp or the current local date/time.

Syntax

```
getdate(timestamp);
```

Parameter : timestamp Specifies an integer Unix timestamp. Default is the current local time (time()) Optional.

Return Value: Returns an associative array with information related to the timestamp

[seconds] - seconds ,[minutes] - minutes, [hours] - hours, [mday] - day of the month, [wday] - day of the week,[mon] - month, [year] - year, [yday] - day of the year, [weekday] - name of the weekday, [month] - name of the month, [0] - seconds since Unix Epoch

PHP getdate() Function

```
<!DOCTYPE html>
<html>
<body>

<?php
// Print the array from getdate()
print_r(getdate());
echo "<br><br>";

// Return date/time info of a timestamp; then format the output
$mydate=getdate(date("U"));
echo "$mydate[weekday], $mydate[month] $mydate[mday],
$mydate[year]";
?>

</body>
</html>
```

PHP getdate() Function

Array ([seconds] => 56 [minutes] => 35 [hours] => 3 [mday] => 24 [wday] => 4 [mon] => 8 [year] => 2017 [yday] => 235 [weekday] => Thursday [month] => August [0] => 1503560156)

Thursday, August 24, 2017

PHP checkdate() Function

- The checkdate() function is used to validate a Gregorian date.

Syntax

```
checkdate(month,day,year);
```

Parameter :

1. **month** Specifies the month as a number between 1 and 12
2. **day** Specifies the day as a number between 1 and 31
3. **year** Specifies the year as a number between 1 and 32767

Return Value: TRUE if the date is valid. FALSE otherwise

PHP checkdate() Function

PHP Scrip

```
<!DOCTYPE html>
<html>
<body>

<?php
var_dump(checkdate(12,31,-400));
echo "<br>";
var_dump(checkdate(2,29,2003));
echo "<br>";
var_dump(checkdate(2,29,2004));
?>

</body>
</html>
```

Output :

```
bool(false)
bool(false)
bool(true)
```

PHP time() Function

- The time() function returns the current time in the number of seconds since the Unix Epoch (January 1 1970 00:00:00 GMT).

Syntax

```
time();
```

Return Value: Returns an integer containing the current time as a Unix timestamp

PHP time() Function

PHP Script :

```
<!DOCTYPE html>
<html>
<body>
<?php
$t=time();
echo($t . "<br>");
echo(date("Y-m-d",$t));
?>
</body>
</html>
```

1503565921
2017-08-24

PHP mktime() Function

- The mktime() function returns the Unix timestamp for a date.
- Tip: This function is identical to mktime() except the passed parameters represents a date (not a GMT date).

Syntax

```
mktime(hour,minute,second,month,day,year,is_dst
  );
```

PHP mktime() Function

```
<!DOCTYPE html>
<html>
<body>
<?php
// Prints: October 3, 1975 was on a Friday
echo "Oct 3, 1975 was on a ".date("l", mktime(0,0,0,10,3,1975)) .
"<br><br>“;
//The mktime() function is useful for doing date arithmetic and validation.
//It will automatically calculate the correct value for out-of-range input:
echo date("M-d-Y",mktime(0,0,0,12,36,2001)) . "<br>";
echo date("M-d-Y",mktime(0,0,0,14,1,2001)) . "<br>";
echo date("M-d-Y",mktime(0,0,0,1,1,2001)) . "<br>";
echo date("M-d-Y",mktime(0,0,0,1,1,99)) . "<br>";
?>
</body>
</html>
```

PHP mktime() Function

Oct 3, 1975 was on a Friday

Jan-05-2002
Feb-01-2002
Jan-01-2001
Jan-01-1999

String Functions

- PHP string function helps us to manipulate string in various ways.
- There are various types of string function available.
- Here we discuss some important functions and its use with examples.

1.echo
2.strtolower ( )
3.strtoupper ( )
4.lcfirst()
5.ucfirst()
6.ucwords()
7.substr_replace ( )
8.More string functions available at https://www.w3schools.com/php/php_ref_string.asp

1. echo: It is used to print one or more string .Echo function is faster than print print function.

Example:

```
<!doctype <!DOCTYPE html>
<html>
<head>
        <title>string function</title>
</head>
<body>
        <?php
        echo "String Function"
        ?>
</body>
</html>
```

Output:

String Function

//echo is not actually function so we are not required to use parentheses with it.

2. strtolower: It converts a string to lower case letter.

Example:

```
<!doctype <!DOCTYPE html>
<html>
<head>
        <title>string function</title>
</head>
<body>
        <?php
        echo strtolower("STRING Function") ;
        ?>
</body>
</html>
```

Output:

string function

3. strtoupper ( ): It converts a string to upper case letter.

Example:

```
<!doctype <!DOCTYPE html>
<html>
<head>
        <title>string function</title>
</head>
<body>
        <?php
        echo strtoupper("String Function") ;
        ?>
</body>
</html>
```

Output:

STRING FUNCTION

4. lcfirst( ): It converts the first character of a string to lower.

Example:

```
<!doctype <!DOCTYPE html>
<html>
<head>
        <title>string function</title>
</head>
<body>
        <?php
        echo lcfirst("String Function") ;
        ?>
</body>
</html>
```

Output:

string Function

5. ucfirst( ): It converts the first character of a string to upper case.

Example:

```
<!doctype <!DOCTYPE html>
<html>
<head>
        <title>string function</title>
</head>
<body>
        <?php
        echo ucfirst("string function") ;
        ?>
</body>
</html>
```

Output:
String function

6. ucwords( ): It converts the first character of every word to upper case.

Example:

```
<!doctype <!DOCTYPE html>
<html>
<head>
        <title>string function</title>
</head>
<body>
        <?php
        echo ucwords("string function") ;
        ?>
</body>
</html>
```

Output:

String Function

7. substr_replace( ): It replaces a part of a string with another string.

The syntax is:
Substr_replace (string, replacement, start , length)

There are various way to replace string. Here the examples are –
Example:

```
<!doctype <!DOCTYPE html>
<html>
<head>
        <title>string function</title>
</head>
<body>
        <?php
        echo substr_replace("hello world","nandon",6);
        ?>
</body>
</html>
```

Output:

hello nandon

//here it replace world to nandon . Number 6 means the replacement starts from 6 position of the string.

Example 2:

```
<!doctype <!DOCTYPE html>
<html>
<head>
        <title>string function</title>
</head>
<body>
        <?php
        echo substr_replace("hello world", "nandon",-5);
        ?>
</body>
</html>
```

Output:

hello nandon

//here in this example we use (-) negative sign so it will count the word from the right hand side and replace.

PHP ltrim() Function

- The ltrim() function removes whitespace or other predefined characters from the left side of a string.
- Related functions:
- rtrim() - Removes whitespace or other predefined characters from the right side of a string.
- trim() - Removes whitespace or other predefined characters from both sides of a string.

Syntax

ltrim(*string,charlist*)

- **Parameter :** String to remove white space from left. Optional. Specifies which characters to remove from the string. If omitted, all of the following characters are removed:"\0" - NULL

"\t" – tab, "\n" - new line, "\x0B" - vertical tab, "\r" - carriage return,

" " - ordinary white space

PHP ltrim() Function

- PHP Script

```
<?php
$str = "Hello World!";
echo $str . "<br>";
echo ltrim($str,"Hello");
?>
```

Output :

```
Hello World!
World!
```

PHP chr() Function

- The chr() function returns a character from the specified ASCII value.
- The ASCII value can be specified in decimal, octal, or hex values. Octal values are defined by a leading 0, while hex values are defined by a leading 0x.

Syntax

chr(*ascii*)

Parameter : An ASCII value to change to character as mandatory.

Return Value :Returns the specified character.

PHP chr() Function

- PHP Script :

```
<?php
echo chr(52) . "<br>"; // Decimal value
echo chr(052) . "<br>"; // Octal value
echo chr(0x52) . "<br>"; // Hex value
?>
```

Output :

```
4
*
R
```

PHP ord() Function

- The ord() function returns the ASCII value of the first character of a string.

Syntax

ord(*string*)

Parameter : The string to get an ASCII value from Character.

Return Value : Returns the ASCII value as an integer

PHP ord() Function

- **PHP Script**

```
<?php
echo ord("h")."<br>";
?>
```

Output :

104

PHP str_replace() Function

- The str_replace() function replaces some characters with some other characters in a string.

Syntax

str_replace(*find,replace,string,count*)

Parameter :

find	Required. Specifies the value to find
replace	Required. Specifies the value to replace the value in *find*
string	Required. Specifies the string to be searched
count	Optional. A variable that counts the number of replacements

Return Value : Returns a string or an array with the replaced value.

- **PHP Script**

```
<?php
echo str_replace("world","Peter","Hello world!");
?>
```

Output :

Hello Peter!

PHP strcmp() Function

- The strcmp() function compares two strings.
- **Note:** The strcmp() function is binary-safe and case-sensitive.

Syntax

strcmp(*string1,string2*)

Parameter : two strings need to pass as input.

Return Value: This function returns:

0 - if the two strings are equal

<0 - if string1 is less than string2

>0 - if string1 is greater than string2

PHP strcmp() Function

- **PHP Script**

```
<?php
echo strcmp("Hello world!","Hello world!");
?>
```

Output :

0

If this function returns 0, the two strings are equal.

PHP str_split() Function

- The str_split() function splits a string into an array.

Syntax

str_split(*string,length*)

Parameter : String that needs to split and length of the array element optional default 1.

Return Value :If length is less than 1, the str_split() function will return FALSE. If length is larger than the length of string, the entire string will be returned as the only element of the array.

PHP str_split() Function

- PHP Script

```
<?php
print_r(str_split("Hello"));
?>
```

Output :

Array ([0] => H [1] => e [2] => l [3] => l [4] => o)

Thank You