What is an Array?

An array is a special variable that can hold many values under a single name, and you can access the values by referring to an index number or name.

The keys of the indexed array are integers that start at 0. Typically, you use indexed arrays when you want to access the elements by their positions.

- **Storing Data in Arrays:**

  An array stores multiple values in one single variable:
- **What is an Array?**

  An array is a special variable, which can hold more than one value at a time.
- **Create an Array in PHP.**

  In PHP, the array() function is used to create an array:

  array();

Example

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

****OUTPUT****

I like Volvo, BMW and Toyota.

- **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);
?>
```

Exercise:

Use the correct function to output the number of items in an array.

$fruits = array("Apple", "Banana", "Orange");

echo _____?

- **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.

1. 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 the index can be assigned manually:

$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

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

******OUTPUT******

I like Volvo, BMW and Toyota.

✓ Processing Arrays with Loops and Iterations an Indexed Array.

✓ To loop through and print all the values of an indexed array, you could use a for loop, like this:

Example

```
<html>
<body>
<?php
$cars = array("Volvo", "BMW", "Toyota");
$arrlength = count($cars);
for($x = 0; $x < $arrlength; $x++) {
 echo $cars[$x];
 echo "<br>";
}
?>
</body>
</html>
```

******OUTPUT******

Volvo

BMW

Toyota

2. 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.";
?>
```

****OUTPUT****

Peter is 35 years old.

Processing Arrays with Loops and Iterations an Associative Array

✓ To loop through and print all the values of an associative array, you could use a foreach loop, like this:

Example

```
<html>
<body>
<?php
$colors = array("red", "green", "blue", "yellow");
foreach ($colors as $value)
        {
          echo "$value <br>";
        }
?>
</body>
</html>
```

****OUTPUT****

red
green
blue

yellow

3. PHP - Multidimensional Arrays:

- ✓ A multidimensional array is an array containing one or more arrays.
- ✓ PHP supports multidimensional arrays that are two, three, four, five, or more levels deep.
- ✓ However, arrays more than three levels deep are hard to manage for most people.

NOTE: The dimension of an array indicates the number of indices you need to select an element.

- ✓ For a two-dimensional array you need two indices to select an element
- ✓ For a three-dimensional array you need three indices to select an element.

- PHP - Two-dimensional Arrays.
  - ✓ A two-dimensional array is an array of arrays (a three-dimensional array is an array of arrays of arrays).
  - ✓ First, take a look at the following table:

Name	Stock	Sold
Volvo	22	18
BMW	15	13
Saab	5	2
Land Rover	17	15

We can store the data from the table above in a two-dimensional array, like this:

```
$cars = array (
array("Volvo",22,18),
array("BMW",15,13),
array("Saab",5,2),
array("Land Rover",17,15)
);
```

Now the two-dimensional $cars array contains four arrays, and it has two indices: row and column.

To get access to the elements of the $cars array we must point to the two indices (row and column):

Example

```
<html>
<body>
<?php
    $cars = array (
    array("Volvo",22,18),
    array("BMW",15,13),
    array("Saab",5,2),
    array("Land Rover",17,15)
    );
echo $cars[0][0].": In stock: ".$cars[0][1].", sold: ".$cars[0][2].".<br>";
echo $cars[1][0].": In stock: ".$cars[1][1].", sold: ".$cars[1][2].".<br>";
echo $cars[2][0].": In stock: ".$cars[2][1].", sold: ".$cars[2][2].".<br>";
echo $cars[3][0].": In stock: ".$cars[3][1].", sold: ".$cars[3][2].".<br>";
?>
</body>
```

```
</html>
```

******OUTPUT******

Volvo: In stock: 22, sold: 18.

BMW: In stock: 15, sold: 13.

Saab: In stock: 5, sold: 2.

Land Rover: In stock: 17, sold: 15.

Processing Arrays with Loops and Iterations an Associative Array

We can also put a for loop inside another for loop to get the elements of the $cars array (we still have to point to the two indices):

```
<html>
<body>
<?php
$cars = array (
 array("Volvo",22,18),
 array("BMW",15,13),
 array("Saab",5,2),
 array("Land Rover",17,15)
 );

for ($row = 0; $row < 4; $row++)
        {
        echo "<p><b>Row number $row</b></p>";
        echo "<ul>";
                for ($col = 0; $col < 3; $col++)
                {
                echo "<li>".$cars[$row][$col]."</li>";
                }
        echo "</ul>";
```

```
        }
?>
</body>
</html>
```

******OUTPUT******

Row number 0
- Volvo
- 22
- 18

Row number 1
- BMW
- 15
- 13

Row number 2
- Saab
- 5
- 2

Row number 3
- Land Rover
- 17
- 15

PHP Sorting Arrays:

PHP - Sort Functions for Arrays

In this chapter, we will go through the following PHP array sort functions:

sort() - sort arrays in ascending order.

rsort() - sort arrays in descending order.

asort() - sort associative arrays in ascending order, according to the value.

ksort() - sort associative arrays in ascending order, according to the key.

arsort() - sort associative arrays in descending order, according to the value.

krsort() - sort associative arrays in descending order, according to the key.

Sort Arrays in Ascending Order - sort()

✓ The following example sorts the elements of the $cars array in ascending alphabetical order:

✓ **Example:**

```
<html>
<body>
<?php
    $cars = array("Volvo", "BMW", "Toyota");
    sort($cars);
    $clength = count($cars);
    for($x = 0; $x < $clength; $x++)
            {
            echo $cars[$x];
            echo "<br>";
            }
?>
</body>
</html>
```

******OUTPUT******

BMW
Toyota
Volvo

✓ The following example sorts the elements of the $numbers array in ascending numerical order:

Example:

```
<html>
<body>
<?php
        $numbers = array(4, 6, 2, 22, 11);
        sort($numbers);
```

```
    $arrlength = count($numbers);
    for($x = 0; $x < $arrlength; $x++)
    {
        echo $numbers[$x];
        echo "<br>";
    }
?>
</body>
</html>
```

******OUTPUT******

2

4

6

11

22

Sort Arrays in descending Order - rsort()

- ✓ The following example sorts the elements of the $cars array in descending alphabetical order:
- ✓ **Example:**

```
<html>
<body>
<?php
    $cars = array("Volvo", "BMW", "Toyota");
    rsort($cars);
    $clength = count($cars);
    for($x = 0; $x < $clength; $x++)
    {
```

```
        echo $cars[$x];
        echo "<br>";
    }
?>
</body>
</html>
```

******OUTPUT******

Volvo

Toyota

BMW

- ✓ The following example sorts the elements of the $numbers array in descending numerical order:

Example:

```
<html>
<body>
<?php
    $numbers = array(4, 6, 2, 22, 11);
    rsort($numbers);
    $arrlength = count($numbers);
    for($x = 0; $x < $arrlength; $x++)
    {
        echo $numbers[$x];
        echo "<br>";
```

```
        }
?>
</body>
</html>
```

****OUTPUT****

22

11

6

4

2

Sort associative arrays in ascending order, according to the value.- asort()

✓ The following example sorts an associative array in ascending order, according to the value:

Example:

```
<html>
<body>
<?php
        $age = array("Peter"=>"39", "Ben"=>"37", "Joe"=>"43");
    asort($age);
    foreach($age as $x => $x_value)
        {
                echo "Key=" . $x . ", Value=" . $x_value;
```

```
            echo "<br>";
        }
    ?>
    </body>
    </html>
```

******OUTPUT******

Key=Peter, Value=35

Key=Ben, Value=37

Key=Joe, Value=43

Sort Array (Ascending Order), According to Key - ksort()

✓ The following example sorts an associative array in ascending order, according to the key:

Example:

```
<html>
<body>
<?php
        $age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");
        ksort($age);
        foreach($age as $x => $x_value)
        {
            echo "Key=" . $x . ", Value=" . $x_value;
            echo "<br>";
        }
?>
</body>
</html>
```

******OUTPUT******

Key=Ben, Value=37

Key=Joe, Value=43

Key=Peter, Value=35

Using Arrays with Forms:-

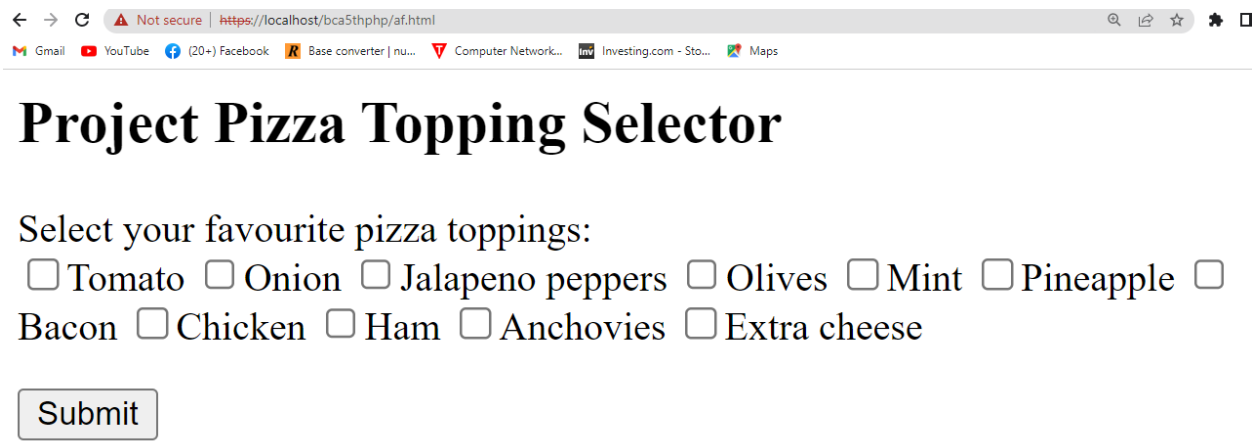
PHP - A Simple HTML Form

Using Arrays with Forms Arrays are particularly potent when used in combination with form elements that support more than one value, such as multiple-selection list boxes or grouped checkboxes. To capture a user's input in an array, simply add square braces to the form element's 'name' to automatically convert it into a PHP array when the form is submitted. The easiest way to illustrate this is with an example. Consider the following form, which holds a multiple-selection list of popular music artists:

Example:

```
<html>
<body>
 <h2>Project Pizza Topping Selector</h2>
 <form method="post" action="pizza.php">
 Select your favourite pizza toppings: <br />
 <input type="checkbox" name="toppings[]" value="tomato">Tomato</input>
 <input type="checkbox" name="toppings[]" value="onion">Onion</input>
 <input type="checkbox" name="toppings[]" value="jalapenos">Jalapeno
peppers</input>
 <input type="checkbox" name="toppings[]" value="olives">Olives</input>
 <input type="checkbox" name="toppings[]" value="mint">Mint</input>
 <input type="checkbox" name="toppings[]" value="pineapple">Pineapple</input>
 <input type="checkbox" name="toppings[]" value="bacon">Bacon</input>
 <input type="checkbox" name="toppings[]" value="chicken">Chicken</input>
 <input type="checkbox" name="toppings[]" value="ham">Ham</input>
 <input type="checkbox" name="toppings[]" value="anchovies">Anchovies</input>
```

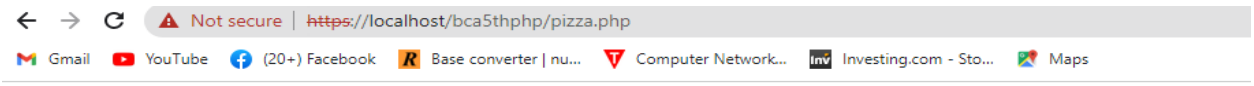
```
<input type="checkbox" name="toppings[]" value="x-cheese">Extra
cheese</input>
<p>
<input type="submit" name="submit" value="Submit" />
</form>
</body>
</html>
```



- Illustrates what the form looks like. And here's the code for the form submission script (pizza.php):

```
<html >
<body>
<h2>Pizza Topping Selector</h2>
You selected the following toppings: <br />
<ul>
<?php
foreach ($_POST['toppings'] as $t) {
echo "<li>$t</li> \r\n";
}
?>
</ul>
```

```
</body>
</html>
```



Pizza Topping Selector

You selected the following toppings:

- tomato
- onion
- pineapple
- x-cheese

Working with Array Functions:

1. PHP array_combine() Function
2. PHP array_keys() Function
3. PHP array_values() Function
4. PHP array_reverse() Function
5. PHP array_change_key_case () Function
6. PHP array_column() Function
7. PHP array_diff() Function
8. PHP array_push() Function
9. PHP array_pop() Function
10. PHP array_product() Function
11. PHP array_sum() Function
12. PHP in_array() Function
13. PHP array_relpace() Function

1) PHP array_combine() Function

Create an array by using the elements from one "keys" array and one "values" array:

Syntax array_combine(*keys*, *values*)

Parameter Values

Parameter	Description
keys	Required. Array of keys
values	Required. Array of values

Example :

```
<html>
<body>
<?php
$name= array("John","Peter","Harry");
$age= array("12","13","14");
$arrcombine = array_combine($name,$age);
print_r($arrcombine);
?>
</body>
</html>
```

Output:

Array ([John] => 12 [Peter] => 13 [Harry] => 14)

2. PHP array_keys() Function

Return an array containing the keys. or The array_keys() function returns an array containing the keys.

Syntax array_keys(*array, value, strict*)

Parameter Values

Parameter	Description
array	Required. Specifies an array
value	Optional. You can specify a value, then only the keys with this value are returned
strict	Optional. Used with the value parameter. Possible values: • true - Returns the keys with the specified value, depending on type: the number 5 is not the same as the string "5". • false - Default value. Not depending on type, the number 5 is the same as the string "5".

Example:

```
<html>
<body>
<?php
// Printing array Keys(not Values)
$a=array("Name"=>"Peter","Age"=>"41","Country"=>"USA");
echo " <b> Array Keys Are :</b><br> <br>";
print_r(array_keys($a));
?>
</body>
</html>
```

Output :

Array Keys Are :

Array ([0] => Name [1] => Age [2] => Country)

3. PHP array_values() Function:

Return all the values of an array (not the keys) or The array_values() function returns an array containing all the values of an array.

Syntax array_values(array);

Example:

```
<html><body>
<?php
```

```
// Printing array values (not Keys)
$a=array("Name"=>"Peter","Age"=>"41","Country"=>"USA");
echo " <b> Array Values Are :</b><br> <br>";
print_r(array_values($a));
?>
</body>
 </html>
```

Output:

Array Values Are :

Array ([0] => Peter [1] => 41 [2] => USA)

4. PHP array_reverse() Function:

Return an array in the reverse order

The array_reverse() function returns an array in the reverse order.

Syntax array_reverse(*array, preserve*)

Parameter Values

Parameter	Description
array	Required. Specifies an array
preserve	Optional. Specifies if the function should preserve the keys of the array or not. Possible values: • true • false

Example

```
<!DOCTYPE html>
<html>
<body>
<?php
$a=array("Volvo","XC90",array("BMW","Toyota"));
$reverse=array_reverse($a);echo"<br>";
$preserve=array_reverse($a,true);echo"<br>";
print_r($a); echo"<br>";
print_r($reverse);echo"<br>";
print_r($preserve);echo"<br>";
```

```
?>
</body>
</html>
```

Output

```
Array ( [0] => Volvo [1] => XC90 [2] => Array ( [0] => BMW [1] => Toyota ) )
Array ( [0] => Array ( [0] => BMW [1] => Toyota ) [1] => XC90 [2] => Volvo )
Array ( [2] => Array ( [0] => BMW [1] => Toyota ) [1] => XC90 [0] => Volvo )
```

5. PHP array_change_key_case () Function:

The array_change_key_case() function changes all keys in an array to lowercase or uppercase.

Syntax : array_change_key_case(*array*, *case*)

Parameter Values

Parameter	Description
array	Required. Specifies the array to use
case	Optional. Possible values: • CASE_LOWER - Default value. Changes the keys to lowercase • CASE_UPPER - Changes the keys to uppercase

Example

```
<html>
<body>
<?php
echo "<b><u>Array_change_key_case() Function</b></u> <br><br> ";
$age=array("PETER"=>"35","BEN"=>"37","JOE"=>"43");
$student=array("name"=>"ram","address"=>"basav circle","city"=>"chikodi", "country"=>
"Indian");
echo "<b>1) Array_change_key_case() Function CASE_LOWER() </b><br><br> ";
print_r(array_change_key_case($age,CASE_LOWER));
```

```
echo"<br><br><br><br>";
echo "<b>2) Array_change_key_case() Function CASE_UPPER()</b> <br><br> ";
print_r(array_change_key_case($student,CASE_UPPER));
?>
</body>
</html>
```

Output

Array_change_key_case() Function

1) Array_change_key_case() Function CASE_LOWER()

Array ([peter] => 35 [ben] => 37 [joe] => 43)

2) Array_change_key_case() Function CASE_UPPER()

Array ([NAME] => ram [ADDRESS] => basav circle [CITY] => chikodi [COUNTRY] => Indian)

6. PHP array_column() Function:

The array_column() function returns the values from a single column in the input array.

Syntax

array_column(*array*, *column_key*, *index_key*)

Parameter Values

Parameter	Description
array	Required. Specifies the multi-dimensional array (record-set) to use. As of PHP 7.0, this can also be an array of objects.
column_key	Required. An integer key or a string key name of the column of values to return. This parameter can also be NULL to return complete arrays (useful together with index_key to re-index the array)
index_key	Optional. The column to use as the index/keys for the returned array

Example:

```
<html>
<body>
<?php
echo "<b><u><center><h2>Array_column() Function</h2></center></b></u> <br><br> ";
$a = array(
 array(
 'id' => 5698,
 'first_name' => 'Peter',
```

```
 'last_name' => 'Griffin',
 ),
 array(
 'id' => 4767,
 'first_name' => 'Ben',
 'last_name' => 'Smith',
 ),
 array(
 'id' => 3809,
 'first_name' => 'Joe',
 'last_name' => 'Doe',
 )
);
$last_names = array_column($a, 'last_name', 'id');
print_r($last_names);
?>
</body>
</html>
```

Output

Array_column() Function

Array ([5698] => Griffin [4767] => Smith [3809] => Doe)

7. PHP array_diff() Function:

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 returns an array that contains the entries from array1 that are not present in array2 or array3, etc.

Syntax

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

Parameter Values

Parameter	Description
array1	Required. The array to compare from
array2	Required. An array to compare against
array3,...	Optional. More arrays to compare against

Example:

```
<html>
<body>
<?php
echo "<b><center><h2>Array_diff() Function on 2 Arrays</h2></center></b><br><br> ";
$a1=array("a"=>"red","b"=>"green","c"=>"blue","d"=>"yellow");
$a2=array("e"=>"red","f"=>"green","g"=>"white");
$result=array_diff($a1,$a2);
print_r($result);
?> </body> </html>
```

Output :

Array_diff() Function on 2 Arrays

Array ([c] => blue [d] => yellow)

8. PHP array_push() Function:

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

Syntax

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

Parameter Values

Parameter	Description
array	Required. Specifies an array
value1	Optional. Specifies the value to add (Required in PHP versions before 7.3)
value2	Optional. Specifies the value to add

Example :

```
<html>
<body>
<?php
echo "<b><center><h2>Array_push() Function </h2></center></b><br><br> ";
$a=array("red","green");
array_push($a,"blue","yellow","pink");
print_r($a);
?>
</body>
</html>
```

Output:

Array_push() Function

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

9. PHP array_pop() Function:

Syntax : array_pop(*array*)

Parameter Values

Parameter	Description
array	Required. Specifies an array

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

Example:

```
<html>
<body>
<?php
echo "<b><center><h2>Array_pop() Function </h2></center></b><br><br> ";
$a=array("red","green","blue","pink","white");
array_pop($a);
print_r($a);
echo"<br>";
array_pop($a);
print_r($a); ?></body></html>
```

Output :

Array_pop() Function

Array ([0] => red [1] => green [2] => blue [3] => pink)

Array ([0] => red [1] => green [2] => blue)

10. PHP array_product() Function:

The array_product() function calculates and returns the product of an array.

Syntax

array_product(*array*)

Parameter Values

Parameter	Description
array	Required. Specifies an array

Example

```
<html>
<body>
<?php
echo "<b><center><h2>Array_product() Function </h2></center></b><br><br> ";
$a=array(5,5);
echo"Product of array(5,5) = ".(array_product($a));
echo"<br>";
$a=array(5,5,2);
echo"Product of array(5,5,2) = ".(array_product($a));
echo"<br>";
$a=array(5,2,1,3);
echo"Product of array(5,2,1,3) = ".(array_product($a));
echo"<br>";
$a=array(25,72,91,0);
echo"Product of array(25,72,91,0) = ".(array_product($a));
echo"<br>";
$a=array(1.1,2.2,3.4);
echo"Product of array(1.1,2.2,3.4) = ".(array_product($a));
echo"<br>";
?>
</body>
</html>
```

Output

Array_product() Function

Product of array(5,5) = 25

Product of array(5,5,2) = 50

Product of array(5,2,1,3) = 30

Product of array(25,72,91,0) = 0

Product of array(1.1,2.2,3.4) = 8.228

11. PHP array_sum() Function:

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

Syntax

array_sum(*array*)

Parameter Values

Parameter	Description
array	Required. Specifies an array

Example:

```
<html>
<body>
<?php
echo "<b><center><h2>Array_sum() Function </h2></center></b><br><br> ";
$a=array(5,5);
echo"Sum of array(5,5) = ".(array_sum($a));
echo"<br>";
$a=array(5,5,2);
echo"Sum of array(5,5,2) = ".(array_sum($a));
echo"<br>";
$a=array(5,2,1,3);
echo"Sum of array(5,2,1,3) = ".(array_sum($a));
echo"<br>";
$a=array(25,72,91,0);
echo"Sum of array(25,72,91,0) = ".(array_sum($a));
echo"<br>";
$a=array(1.1,2.2,3.4);
```

```
echo"Sum of array(1.1,2.2,3.4) = ".(array_sum($a));
echo"<br><br>";
echo"<b>Sum on Associative Array</b></br>";
$a=array("a"=>52.2,"b"=>13.7,"c"=>0.9);
echo"Sum of array(52.2,13.7,0.9) = ".(array_sum($a));
?> </body></html>
```

Output :

Array_sum() Function
Sum of array(5,5) = 10
Sum of array(5,5,2) = 12
Sum of array(5,2,1,3) = 11
Sum of array(25,72,91,0) = 188
Sum of array(1.1,2.2,3.4) = 6.7
Sum on Associative Array
Sum of array(52.2,13.7,0.9) = 66.8

12. PHP in_array() Function:

The in_array() function searches an array for a specific value.

Note: 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 Values

Parameter	Description
search	Required. Specifies the what to search for
array	Required. Specifies the array to search
type	Optional. If this parameter is set to TRUE, the in_array() function searches for the search-string and specific type in the array.

Example

```
<html>
<body>
<?php
echo "<b><center><h2>in_array() Function </h2></center></b><br><br> ";
$people = array("Peter", "Joe", "Glenn", "Cleveland");
if (in_array("Glenn", $people))
 {
 echo "Match found";
 }
```

```
else
 {
 echo "Match not found";
 }
?> </body> </html>
```

Output :

in_array() Function

Match found

13. PHP array_relpace() Function

The array_replace() function replaces the values of the first array with the values from following arrays.

Syntax

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

Parameter Values

Parameter	Description
array1	Required. Specifies an array
array2	Optional. Specifies an array which will replace the values of *array1*
array3,...	Optional. Specifies more arrays to replace the values of *array1* and *array2, etc.* Values from later arrays will overwrite the previous ones.

Example:

```
<html>
<body>
<?php
echo "<b><center><h2>Array_replace () Function </h2></center></b><br><br> ";
$a1=array("red","green");
$a2=array("blue","yellow");
echo"Replaced Array Values are :<br> ";
print_r(array_replace($a1,$a2));
echo"<br><br>";
$a1=array("a"=>"red","b"=>"green","c"=>"pink");
$a2=array("a"=>"orange","burgundy","yellow");
echo"Replaced Array Values are :<br> ";
```

```
print_r(array_replace($a1,$a2));
echo"<br><br>";
$a1=array("red","green");
$a2=array("blue","yellow");
$a3=array("orange","burgundy");
echo"Replaced Array Values are :<br> ";
print_r(array_replace($a1,$a2,$a3));
?></body></html>
```

Output :

Array_replace () Function

Replaced Array Values are :

Array ([0] => blue [1] => yellow)

Replaced Array Values are :

Array ([a] => orange [b] => green [c] => pink [0] => burgundy [1] => yellow)

Replaced Array Values are :

Array ([0] => orange [1] => burgundy)

Working with Dates and Times:

The PHP date() function formats a timestamp to a more readable date and time.

Syntax

```
date(format,timestamp)
```

Parameter	Description
format	Required. Specifies the format of the timestamp
timestamp	Optional. Specifies a timestamp. Default is the current date and time

Get a Date:

The required format parameter of the date() function specifies how to format the date (or time).

Here are some characters that are commonly used for dates:

d - Represents the day of the month (01 to 31)

m - Represents a month (01 to 12)

Y - Represents a year (in four digits)

l (lowercase 'L') - Represents the day of the week

Other characters, like"/", ".", or "-" can also be inserted between the characters to add additional formatting.

The example below formats today's date in three different ways:

```
<html>
<body>
<?php
echo "Today is " . date("Y/m/d") . "<br>";
echo "Today is " . date("Y.m.d") . "<br>";
echo "Today is " . date("Y-m-d") . "<br>";
echo "Today is " . date("l");
?>
</body>
</html>
```

OUTPUT:

Today is 2020/11/03
Today is 2020.11.03
Today is 2020-11-03
Today is Tuesday

Get a Time:

Here are some characters that are commonly used for times:

H - 24-hour format of an hour (00 to 23)

h - 12-hour format of an hour with leading zeros (01 to 12)

i - Minutes with leading zeros (00 to 59)

s - Seconds with leading zeros (00 to 59)

a - Lowercase Ante meridiem and Post meridiem (am or pm)

The example below outputs the current time in the specified format:

```
<html>
<body>
<?php
```

```
echo "The time is " . date("h:i:sa");
?>
</body>
</html>
```

OUTPUT:

The time is 07:25:37am

Important Questions:

2 Marks

1. What is Associative Array? Give simple example.
2. Which function is used to get length of an array. Give simple example.
3. What are PHP string functions? List any 5.
4. With an example explain floor and ceil.
5. Write a PHP program to add and remove Array elements.
6. Write a PHP program for checking Date validity.

5 and 10 Marks

1. List and explain different Types of Arrays in PHP.
2. Explain arrays with forms with an Example.
3. Explain multidimensional arrays in PHP with example
4. List and explain different characters that are used to specify date and time.
5. Define PHP arrays. Write Syntax to create PHP arrays and give simple Example.
6. Explain PHP numeric functions with example.
7. List and explain Different Sorting functions that are present in PHP.