

Course 13: Web Applications Development using PHP & MYSQL

UNIT – II

Arrays

An **array** in PHP is a variable that can store **multiple values in a single variable**. Each value is identified by an **index** or **key**.

Advantages of PHP Array

- **Lesser code:** Feature grouping takes precedence over multiple variables because arrays can accommodate multiple values within a single structure thus simplifying code options and reducing repetition.
- **Easy to traverse:** Using arrays enables developers to move through data elements inside loops which eliminates direct variable handling requirements.
- **Ability to sort:** PHP includes multiple sorting functions that help users easily organise their array data.
- **Dynamic memory allocation:** The storage capacity of arrays grows automatically while new data items are added which provides adaptive memory features.
- **Data grouping:** When grouping related or similar data together the selection and retrieval operations become simpler

PHP mainly supports:

1. **Indexed arrays** – use numeric indexes.
2. **Associative arrays** – use named keys.
3. **Multidimensional arrays** – arrays containing other arrays.

PHP Indexed array

An indexed array stores values with numeric indexes starting from 0

Syntax:

```
$season = array("summer", "winter", "spring", "autumn");
```

Example:

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

Output:

Seasons are: summer, winter, spring and autumn

Associative Array

An associative array uses **key-value pairs**.

Syntax:

```
$salary = array("Jonas" => "350000", "John" => "450000", "Karl" => "200000");
```

Example:

```
<?php  
  
$salary = array("Jonas" => "350000", "John" => "450000", "Karl" => "200000");  
  
echo "Jonas salary: ".$salary["Jonas"]."<br/>";  
  
echo "John salary: ".$salary["John"]."<br/>";  
  
echo "Karl salary: ".$salary["Karl"]."<br/>";  
  
?>
```

Output:

Jonas salary: 350000

John salary: 450000

Karl salary: 200000

PHP Multidimensional Array

A multidimensional array exists when one or multiple arrays are included within it.

Syntax:

```
$emp = array  
  
(  
    array(1,"Sam",400000),  
    array(2,"John",500000),  
    array(3,"Rick",300000)  
);
```

Example:

```
<?php  
  
$emp = array  
  
(  
    array(1,"Sam",400000),
```

```

    array(2,"John",500000),
    array(3,"Rick",300000)
);

for ($row = 0; $row < 3; $row++) {
    for ($col = 0; $col < 3; $col++) {
        echo $emp[$row][$col]." ";
    }
    echo "<br/>";
}

?>

```

Output:

```

1 Sam 400000
2 John 500000
3 Rick 300000

```

Array-Related Functions

PHP provides many built-in functions for working with arrays.

Arrays in PHP are used to store and manage multiple values. We can **create, add, remove, search, count, sort, and reverse** array elements using PHP's built-in array functions.

Function	Purpose	Example
count()	Finds number of elements	count(\$a)
sort()	Sorts array in ascending order	sort(\$a)
rsort()	Sorts in descending order	rsort(\$a)
array_push()	Adds elements at the end	array_push(\$a, "C")
array_pop()	Removes last element	array_pop(\$a)
array_shift()	Removes first element	array_shift(\$a)

Function	Purpose	Example
array_unshift()	Adds element at beginning	array_unshift(\$a, "Java")
in_array()	Checks whether a value exists	in_array("PHP",\$a)
array_reverse()	Reverses an array	array_reverse(\$a)

PHP String

PHP string is a sequence of characters i.e., used to store and manipulate text.

here are 4 ways to specify a string literal in PHP.

1. **Single Quoted**
2. **Double Quoted**
3. **Heredoc Syntax**
4. **Newdoc Syntax**

Single Quoted

We can create a string in PHP by enclosing the text in a single-quote. It is the easiest way to specify string in PHP.

Example:

```
<?php
    $str='Hello text within single quote';
    echo $str;
?>
```

Output:

Hello text within single quote

Example:

```
$str2='Using double "quote" directly inside single quoted string';
$str3='Using escape sequences \n in single quoted string';
echo $str2;
```

```
echo $str3;
```

Output:

Using double "quote" directly inside single quoted string

Using escape sequences \n in single quoted string

Double Quoted

In PHP, we can specify string through enclosing text within double quote also. But escape sequences and variables will be interpreted using double quote PHP strings.

Example:

```
<?php
```

```
$str="Hello text within double quote";
```

```
echo $str;
```

```
?>
```

Output:

Hello text within double quote

Example2:

```
$str2="Using double \"quote\" with backslash inside double quoted string";
```

```
$str3="Using escape sequences \n in double quoted string
```

```
echo $str2;
```

```
echo $str3;
```

Output:

Using double "quote" with backslash inside double quoted string

Using escape sequences in double quoted string

Heredoc

Heredoc syntax (<<<) is the third way to delimit strings. In Heredoc syntax, an identifier is provided after this heredoc <<< operator, and immediately a new line is started to write any text.

Example

```
<?php
```

```
$str = <<<Demo
```

It is a valid example

```
Demo; //Valid code as whitespace or tab is not valid before closing identifier
```

```
echo $str;
```

```
?>
```

Output:

It is a valid example

Newdoc

Newdoc is similar to the heredoc, but in newdoc parsing is not done. It is also identified with three less than symbols <<< followed by an identifier.

But here identifier is enclosed in single-quote, e.g. <<<'EXP'. Newdoc follows the same rule as heredocs.

PHP String functions

PHP string functions allow text manipulation, which is generally referred to as "strings". They provide formatting, searching, and altering string data easily.

Common PHP String Functions

Function	Purpose	Example
strlen()	Finds the length of a string	strlen("Hello")
strtoupper()	Converts string to uppercase	strtoupper("hello")
strtolower()	Converts string to lowercase	strtolower("HELLO")
ucfirst()	Converts first character to uppercase	ucfirst("hello")
strrev()	Reverses a string	strrev("Hello")
strpos()	Finds the position of a substring	strpos("Hello PHP", "PHP")
str_replace()	Replaces text	str_replace("PHP", "Java", \$s)

Function	Purpose	Example
substr()	Extracts part of a string	substr("Hello",1,3)
strcmp()	Compares two strings	strcmp("abc","abc")
trim()	Removes spaces from beginning and end	trim(\$s)
explode()	Converts string into an array	explode(" ", \$s)
implode()	Joins array elements into a string	implode(" ", \$a)

Dates and Time in PHP

PHP provides many built-in functions for working with **dates and time**. These functions are useful for displaying the current date, calculating dates, formatting dates, comparing dates, and manipulating time values.

Formatting Strings with PHP

Syntax:

date(format);

Example:

```
<?php
echo date("d-m-Y");

?>
```

Output:

17-09-2026

Format	Meaning	Example
d	Day of month (01–31)	17
m	Month (01–12)	09
Y	Four-digit year	2026
y	Two-digit year	26

Format	Meaning	Example
l	Full day name	Thursday
D	Short day name	Thu
F	Full month name	September
M	Short month name	Sep

Different Date Formats

```
<?php
```

```
echo date("d-m-Y");
```

```
echo "<br>";
```

```
echo date("d/m/Y");
```

```
echo "<br>";
```

```
echo date("Y-m-d");
```

```
echo "<br>";
```

```
echo date("l, F d, Y");
```

```
?>
```

Possible output:

17-09-2026

17/09/2026

2026-09-17

Thursday, September 17, 2026

Formatting Time

Format	Meaning	Example
H	24-hour format	14
h	12-hour format	02
i	Minutes	30
s	Seconds	45
a	am/pm	pm
A	AM/PM	PM

```
<?php
echo date("H:i:s");
echo "<br>";
echo date("h:i:s A");
?>
```

Output might be:

14:30:45

02:30:45 PM

Manipulating Dates and Time with PHP

PHP allows us to **add or subtract days, months, years, hours, etc.**

The `strtotime()` function converts an English textual date/time description into a Unix timestamp.

Example 1: Tomorrow

```
<?php
echo date("d-m-Y", strtotime("+1 day"));
?>
```

If today is 17-09-2026, output:

18-09-2026

Example 2: Yesterday

```
<?php
echo date("d-m-Y", strtotime("-1 day"));
?>
```

Output:

16-09-2026

Example3: After 7 Days

```
<?php
echo date("d-m-Y", strtotime("+7 days"));
?>
```

Output:

24-09-2026

Using DateTime Class

PHP also provides the DateTime class for more powerful date and time manipulation.

Example

```
<?php
```

```
$date = new DateTime("2026-09-17");
```

```
echo $date->format("d-m-Y");
```

```
?>
```

Output:

17-09-2026

Working with Objects in PHP

PHP supports **Object-Oriented Programming (OOP)**. An **object** is an instance of a class. A class defines the properties (data) and methods (functions) that an object can have.

For example, a Student class can contain student details such as name, roll number, and course, along with methods such as display().

Creating Objects in PHP

An object is created from a class using the new keyword.

Syntax

```
$ObjectName = new ClassName();
```

Accessing Object Instances

The **arrow operator (->)** is used to access the properties and methods of an object.

Example

```
<?php
```

```
class Student
```

```
{
```

```
    public $name;
```

```
    public $rollno;
```

```
public function display()
{
    echo "Name: " . $this->name . "<br>";
    echo "Roll Number: " . $this->rollno;
}
}
```

```
$student1 = new Student();
$student1->name = "Ravi";
$student1->rollno = 101;
$student1->display();
?>
```

Output

Name: Ravi

Roll Number: 101