INTRODUCTION TO PHP

- Server-Side Scripting
- MySQL
- PHP
- Apache
- Variables
- Control Structure
- Looping

Server-Side Scripting

- A script is a set of programming instructions that is interpreted at runtime.
- The purpose of the scripts is usually to enhance the performance or perform routine tasks for an application.
  - Client-side
  - Server-side
- In server-side scripting, (such as PHP, ASP) the script is processed by the server Like: Apache, ColdFusion, ISAPI and Microsoft's IIS on Windows.
- Client-side scripting such as JavaScript runs on the web browser.

Server-Side Scripting – Continued

- Advantages of Server-Side Scripting
  - Dynamic content.
  - Computational capability.
  - Database and file system access.
  - Network access (from the server only).
  - Built-in libraries and functions.
  - Known platform for execution (as opposed to client- side, where the platform is uncontrolled.)
  - Security improvements

Introduction to PHP

What is PHP?

- PHP stands for PHP: Hypertext Preprocessor
- Developed by Rasmus Lerdorf in 1994
- It is a powerful server-side scripting language for creating dynamic and interactive websites.
- It is an open source software, which is widely used and free to download and use (PHP is FREE to download from the official PHP resource: www.php.net).
- It is an efficient alternative to competitors such as Microsoft's ASP.

Introduction to PHP

PHP scripts are executed on the server

- PHP supports many databases (MySQL, Informix, Oracle, Sybase, Solid, PostgreSQL, Generic ODBC, etc.)
- PHP is an open source software
- PHP is free to download and use

Introduction to PHP

- PHP is perfectly suited for Web development and can be embedded directly into the HTML code.
- The PHP syntax is very similar to JavaScript, Perl and C.
- PHP is often used together with Apache (web server) on various operating systems. It also supports ISAPI and can be used with Microsoft's IIS on Windows.
- PHP supports many databases (MySQL, Informix, Oracle, Sybase, Solid, PostgreSQL, Generic ODBC, etc.)

Introduction to PHP

- What is a PHP File?
- PHP files may contain text, HTML tags and scripts
- PHP files are returned to the browser as plain HTML
- PHP files have a file extension of ".php", ".php3", or ".phtml“
- What is MySQL?
  - MySQL is a database server
  - MySQL is ideal for both small and large applications
  - MySQL supports standard SQL
  - MySQL compiles on a number of platforms
  - MySQL is free to download and use

Introduction to PHP

- **What you need to develop PHP Application:**
  - Install Apache (or IIS) on your own server, install PHP, and MySQL

    OR
  - Install Wampserver2 (a bundle of PHP, Apache, and MySql server)   on your own server/machine

Advantages of PHP

- Easy to learn: PHP is easy to learn and use. For beginner programmers who just started out in web development, PHP is often considered as the preferable choice of language to learn.
- Open source: PHP is an open-source project. It is developed and maintained by a worldwide community of developers who make its source code freely available to download and use.
- Portability: PHP runs on various platforms such as Microsoft Windows, Linux, Mac OS, etc. and it is compatible with almost all servers used today such Apache, IIS, etc.

Advantages of PHP

- Fast Performance: Scripts written in PHP usually execute or runs faster than those written in other scripting languages like ASP, Ruby, Python, Java, etc.
- Vast Community: Since PHP is supported by the worldwide community, finding help or documentation related to PHP online is extremely easy.

PHP Installation Downloads

Free Download

- PHP: http://www.php.net/downloads.php
- **MySQL Database**: http://www.mysql.com/downloads/index.html
- **Apache Server:** http://httpd.apache.org/download.cgi

- How to install and configure apache
- Here is a link to a good tutorial from PHP.net on how to install PHP5: http://www.php.net/manual/en/install.php

How to Download & Install XAMPP on Windows

- What is XAMPP?
- XAMPP is an open-source, cross-platform web server that consists of a web server, MySQL database engine, and PHP and Perl programming packages. It is compiled and maintained by Apache.
- It allows users to create WordPress websites online using a local web server on their computer. It supports Windows, Linux, and Mac.
- It is compiled and maintained by apache

How to Download & Install XAMPP on Windows

- The acronym XAMPP stands for;
- X – [cross platform operating systems] meaning it can run on any OS Mac OX , Windows , Linux etc.
- A – Apache - this is the web server software.
- M – MySQL - Database.
- P – PHP
- P – Perl – scripting language

How PHP is Processed

- When a PHP document is requested of a server, the server will send the document first to a PHP processor
- Two modes of operation
  - **Copy mode** in which plain HTML is copied to the output
  - **Interpret mode** in which PHP code is interpreted and the output from that code sent to output
  - The client never sees PHP code, only the output produced by the code

Basic PHP Syntax

- The PHP script can be embedded within HTML web pages.
- A PHP script starts with the <?php and ends with the ?> tag.
- The PHP delimiter <?php and ?> in the following example simply tells the PHP engine to treat the enclosed code block as PHP code, rather than simple HTML.

```
<?php
// Some code to be executed
echo "Hello, world!";
?>
```

Basic PHP Syntax

- Embedding PHP within HTML
- PHP files are plain text files with .php extension.
- Inside a PHP file you can write HTML like you do in regular HTML pages as well as embed PHP codes for server side execution.

```
<html>  <head><title>A  Simple  PHP  File</title>
</head>
 <body>
 <h1><?php echo "Hello, world!"; ?></h1> </body>
 </html>
```

- when you run this code the PHP engine executed the instructions between the <?php … ?> tags and leave rest of the thing as it is.
- At the end the web server send the final output back to your browser which is completely in HTML.

Basic PHP Syntax

- **PHP statements are terminated with semicolons ;**
- Curly braces, { } are used to create compound statements
- Variables cannot be defined in a compound statement unless it is the body of a function
- PHP has typical scripting language characteristics
  - Dynamic typing, un-typed variables
  - Associative arrays
  - Pattern matching
  - Extensive libraries
- **Primitives, Operations, Expressions**
  - Four scalar types: boolean, integer, double, string
  - Two compound types: array, object
  - Two special types: resource and NULL

Basic PHP Syntax

- A PHP scripting block always starts with <?php and ends with ?>

<?php ………………. ?>

– Other options are:

1. <? ………………… ?>
2. <script> ... </script>

- There are three basic statements to output text with PHP:

echo, print, and printf.

Example: echo 'This is a <b>test</b>!';

PHP Comments

- A comment in PHP code is a line that is not executed as a part of the program.
- Its only purpose is to be read by someone who is looking at the code.

- <?php
- // This is a single-line comment

- # This is also a single-line comment

- /*
- This is a multiple-lines comment block
- that spans over multiple
- lines
- */
- ?>

Basic PHP Syntax

Example 1

```
<html >
<head> <title>Simple PHP Example</title>
<body>
<?php
echo "Hello Class of 2021. This is my first PHP Script";  echo "<br />";
print "<b><i>What have you
        learnt and how many friends have you made?</i></b>";
?>
</body>
</html>
```

Echo and Print Statements

- Echo
- The echo statement can output one or more strings.
- echo can be used with or without parentheses
  e.g. echo or echo().
- echo does not return any value.
- We can pass multiple strings separated by comma (,) in echo.
- echo is faster than print statement.

Echo and Print Statements

- `<?php`
- `// Displaying string of text`
  `echo "Hello World!";`
- `?>`

Echo and Print Statements

- Display HTML Code
- <?php
- // Displaying HTML code
- echo "<h4>This is a simple heading.</h4>";
- echo "<h4 style='color: red;'>This is heading with style.</h4>";
- ?>

Output : **This is a simple heading.**

This is heading with style.

Echo and Print Statements

- The print statement (an alternative to echo) is set to display output to the browser.
- print can be used with or without parentheses.
- Using print, we cannot pass multiple arguments.
- print is slower than echo statement.
- Both echo and print statement works exactly the same way except that the print statement can only output one string, and always returns 1.
- That's why the echo statement considered marginally faster than the print statement since it doesn't return any value.

Echo and Print Statements

- Display Strings of Text
- <?php
- // Displaying string of text
- print "Hello World!";
- ?>
- Display HTML Code
- <?php // Displaying HTML code print “
- <h4>This is a simple heading.</h4>";
- print "<h4 style='color: red;'>This is heading with style.</h4>";
- ?>

Variables

- Variables are used to store data, like string of text, numbers, etc.
- Variable values can change over the course of a script.
- Here're some important things to know about variables:
- In PHP, a variable does not need to be declared before adding a value to it.
- PHP automatically converts the variable to the correct data type, depending on its value.
- After declaring a variable it can be reused throughout the code.
- The assignment operator (=) used to assign value to a variable.

Variables

- Example

```
<?php
// Declaring variables
$txt = "Hello World!";
$number = 10;

// Displaying variables value
echo $txt;  // Output: Hello World!
echo $number; // Output: 10
?>
```

Variable Naming Rules

- All variables in PHP start with a $ sign, followed by the name of the variable.
- A variable name must start with a letter or the underscore character _.
- A variable name cannot start with a number.
- A variable name in PHP can only contain alpha-numeric characters and underscores (A-z, 0-9, and _).
- A variable name cannot contain spaces

PHP Data Types

- PHP supports total eight primitive data types:
  - Integer
  - Floating point number or Float
  - String
  - Booleans
  - Array
  - Object
  - Resource
  - NULL.
  - These data types are used to construct variables.

PHP Data Types

- Integer
- Integer means numeric data with a negative or positive sign.
- It holds only whole numbers, i.e., numbers without fractional part or decimal points.
- Rules for integer:

1. An integer can be either positive or negative.
2. An integer must not contain decimal point.
3. Integer can be decimal (base 10), octal (base 8), or hexadecimal (base 16).
4. The range of an integer must be lie between 2,147,483,648 and 2,147,483,647 i.e., -2^31 to 2^31

PHP Data Types

```
<?php
    $dec1 = 34;
     $oct1 = 0243;
      $hexa1 = 0x45;
      echo "Decimal number: " .$dec1. "</br>";
      echo "Octal number: " .$oct1. "</br>";
      echo   "HexaDecimal   number:   "   .$hexa1.
"</br>";
?>
```

OutPut

```
Decimal number: 34
Octal number: 163
HexaDecimal number: 69
```

:

PHP Data Types

- Float
- A floating-point number is a number with a decimal point.
- Unlike integer, it can hold numbers with a fractional or decimal point, including a negative or positive sign.
- <?php
-     $n1 = 19.34;
-     $n2 = 54.472;
-     $sum = $n1 + $n2;
-     echo "Addition of floating numbers: " .$sum;
- ?> **Output** Addition of floating numbers: 73.812

:

PHP Data Types

- Strings
- Strings are sequences of characters, where every character is the same as a byte.
- A string can hold letters, numbers, and special characters and it can be as large as up to 2GB (2147483647 bytes maximum).
- The simplest way to specify a string is to enclose it in single quotes (e.g. 'Hello world!'), however you can also use double quotes ("Hello world!").

:

PHP Data Types

- Strings
- Strings are sequences of characters, where every character is the same as a byte.
- A string can hold letters, numbers, and special characters and it can be as large as up to 2GB (2147483647 bytes maximum).
- The simplest way to specify a string is to enclose it in single quotes (e.g. 'Hello world!'), however you can also use double quotes ("Hello world!").

:

PHP Data Types

- Strings
- <?php
- $a = 'Hello world!';
- echo $a;
- echo "<br>";
-
- $b = "Hello world!";
- echo $b;
- echo "<br>";
-
- ?>

:

PHP Data Types

- Boolean
- Hold only two values, either TRUE or FALSE.
- Successful events will return true and unsuccessful events return false. NULL type values are also treated as false in Boolean.
- Apart from NULL, 0 is also consider as false in boolean.
- If a string is empty then it is also considered as false in boolean data type.
- $x = true;
- $y = false;

:

PHP Data Types

- Arrays
- An array is a variable that can hold more than one value at a time.
- It is useful to aggregate a series of related items together, for example a set of country or city names.
- An array is formally defined as an indexed collection of data values.
- Each index (also known as the key) of an array is unique and references a corresponding value.

:

PHP Data Types

- Arrays
- <?php
-   $intArray = array(10, 20 , 30);
-   echo "First Element: $intArray[0]\n";
  echo "Second Element: $intArray[1]\n";
- echo "Third Element: $intArray[2]\n";
- ?>
- Output\
- First Element: 10
- Second Element: 20
- Third Element: 30

:

PHP Data Types

- Objects
- An object is a data type that not only allows storing data but also information on, how to process that data.
- An object is a specific instance of a class which serve as templates for objects.
- Objects are created based on this template via the new keyword.
- Every object has properties and methods corresponding to those of its parent class.
- Every object instance is completely independent, with its own properties and methods, and can thus be manipulated independently of other objects of the same class.

PHP Data Types

- Objects

```
<?php
     class bike {
          function model() {
               $model_name = "Royal Enfield";
               echo "Bike Model: " .$model_name;
             }
     }
     $obj = new bike();
     $obj -> model();
?>
```

- Output:
- Bike Model: Royal Enfield

:

PHP var_dump() function

- The var_dump() function is a built-in function of PHP that dumps the information about the variables.
- This information includes the data type and value of the variable. In case of string, it also includes the size of the string passed inside the function.

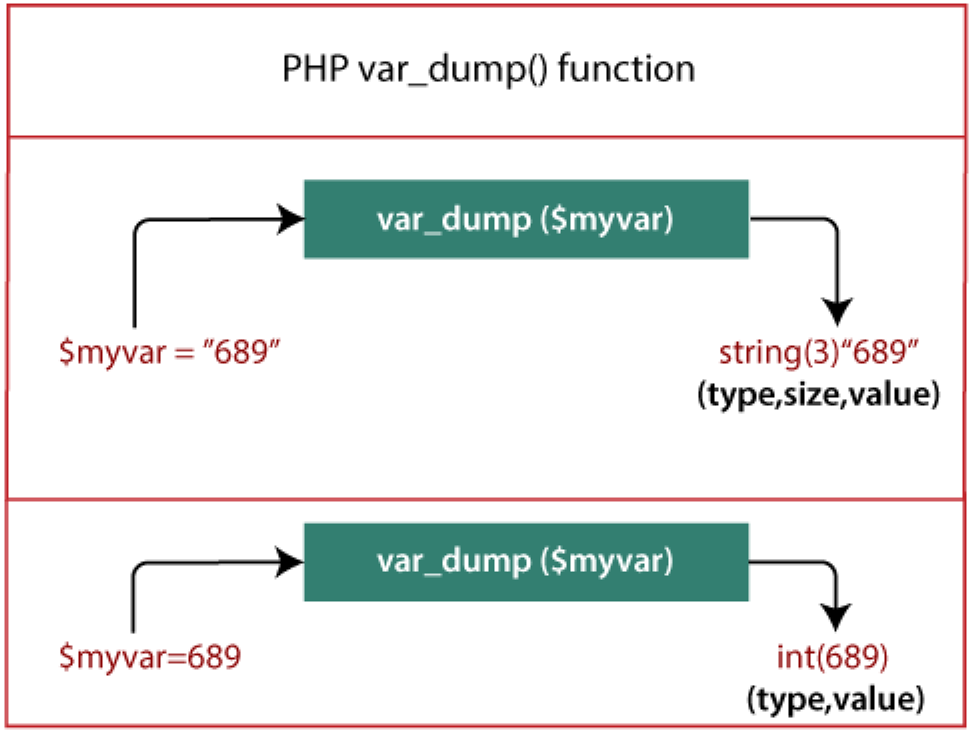
- The array and object are explored recursively with values to show their structure.

- In simple words, this function provides structured information about one or more variables.
- Syntax
- var_dump(var1, var2, ...);

:

PHP var_dump() function

- Parameter
- Expression (var1, var2, ...): variable or the value of the variable, which you want to dump.



:

PHP var_dump() function

- <?php
- //PHP program to demonstrate the working of var_dump function
- $x = 25;
- //dump integer variable
- var_dump ($x);
- echo "</br>“$msg1 = "Hello Alex";
- var_dump ($msg1);
- ?>

:

PHP Data Types

- Resources
- A resource is a special variable, holding a reference to an external resource.
- Resource variables typically hold special handlers to opened files and database connections.
- <?php
- // Open a file for reading
- $handle = fopen("note.txt", "r");
- var_dump($handle);
- echo "<br>";
- link = mysql_connect("localhost", "root", "");
- var_dump($link);
- ?>

:

PHP Data Types

- NULL
- The special NULL value is used to represent empty variables in PHP.
- A variable of type NULL is a variable without any data. NULL is the only possible value of type null.

```
<?php
    $nl = NULL;
    echo $nl;   //it will not give any output
?>
```

PHP Operators

- Arithmetic Operators
- The arithmetic operators are used to perform common arithmetical operations, such as addition, subtraction, multiplication etc. Here's a complete list of PHP's arithmetic operators:

Operator	Description	Example	Result
+	Addition	$x + $y	Sum of $x and $y
-	Subtraction	$x - $y	Difference of $x and $y.
*	Multiplication	$x * $y	Product of $x and $y.
/	Division	$x / $y	Quotient of $x and $y
%	Modulus	$x % $y	Remainder of $x divided by $y

Assignment Operators

- The assignment operators are used to assign values to variables.

Operator	Description	Example	Is The Same As
=	Assign	$x = $y	$x = $y
+=	Add and assign	$x += $y	$x = $x + $y
-=	Subtract and assign	$x -= $y	$x = $x - $y
*=	Multiply and assign	$x *= $y	$x = $x * $y
/=	Divide and assign quotient	$x /= $y	$x = $x / $y
%=	Divide and assign modulus	$x %= $y	$x = $x % $y

Comparison Operators

- The comparison operators are used to compare two values in a Boolean fashion.

Operator	Name	Example	Result
==	Equal	$x == $y	True if $x is equal to $y
===	Identical	$x === $y	True if $x is equal to $y, and they are of the same type
!=	Not equal	$x != $y	True if $x is not equal to $y
<>	Not equal	$x <> $y	True if $x is not equal to $y
!==	Not identical	$x !== $y	True if $x is not equal to $y, or they are not of the same type
<	Less than	$x < $y	True if $x is less than $y
>	Greater than	$x > $y	True if $x is greater than $y
>=	Greater than or equal to	$x >= $y	True if $x is greater than or equal to $y
<=	Less than or equal to	$x <= $y	True if $x is less than or equal to $y

Incrementing and Decrementing Operators

- The increment/decrement operators are used to increment/decrement a variable's value.

Operator	Name	Effect
++$x	Pre-increment	Increments $x by one, then returns $x
$x++	Post-increment	Returns $x, then increments $x by one
--$x	Pre-decrement	Decrements $x by one, then returns $x
$x--	Post-decrement	Returns $x, then decrements $x by one

Logical Operators

- The logical operators are typically used to combine conditional statements.

Operator	Name	Example	Result
and	And	$x and $y	True if both $x and $y are true
or	Or	$x or $y	True if either $x or $y is true
xor	Xor	$x xor $y	True if either $x or $y is true, but not both
&&	And	$x && $y	True if both $x and $y are true
||	Or	$x || $y	True if either $$x or $y is true
!	Not	!$x	True if $x is not true

== versus ===

- Two “equality” operators
  - == tests for “equality” in value but not necessarily type
  - === tests for “identity” in value AND type
- == ignores the distinction between:
  - Integers, floating point numbers, and strings containing the same numerical value
  - Nonzero numbers and boolean TRUE
  - Zero and boolean FALSE
  - Empty string, the string ‘0’ and boolean FALSE
  - Any other non-empty string and boolean TRUE

Strings

- A sequence of characters
- Single and double quotes:
  - Suppose `$str = 42;`
  - `echo 'With single quotes, str is $str';`

    → **output:** `With single quotes, str is $str`
  - `echo "With double quotes, str is $str";`

    → **output:** `With double quotes, str is 42`

Strings in PHP

a string is a sequence of letters, symbols, characters and arithmetic values or combination of all tied together in single or double quotes.

- String literals are enclosed in single or double quotes
- Example:

```
<?php
$sum = 20;
echo 'the sum is:
$sum';  echo "<br
/>";
echo "the sum is: $sum";
echo "<br />";
echo '<input type="text" name="first_name" id="first_name">';
?>
```

- Double quoted strings have escape sequences (such as /n or /r) interpreted and variables interpolated (substituted)
- Single quoted strings have neither escape sequence interpretation nor variable interpolation
- A literal $ sign in a double quoted string must be escaped with a backslash, \

The Concatenation Operator

- The concatenation operator (.) is used to put two string values together.
- Example:

```
<?php
  $txt1="Hello Everyone,";
  $txt2="1234 is Dan’s home address";
  echo $txt1.$txt2;
?>
```

Control Structure

- Control structures are the building blocks of any programming language. PHP provides all the control structures that you may have encountered anywhere. The syntax is the same as C or Perl.
- Making computers think has always been the goal of the computer architect and the programmer. Using control structures computers can make simple decisions and when programmed cleverly they can do some complex things.

Conditional Statements

1. The If...Else Statement

Syntax

if (*condition*) *code to be executed if condition is true;*

else *code to be executed if condition is false;*

```
<?php
$d=date("D");
if ($d=="Fri") echo "Have a nice weekend!";
else echo "Have a nice day!";
?>
```

If more than one line should be executed if a condition is true/false, the lines should be enclosed within curly braces:

```
<?php
$num=12;
if($num<100){
echo "$num is less than 100";
}
?>
```

Conditional Statements

2.The ElseIf Statement

- If you want to execute some code if one of several conditions is true use the elseif statement

Syntax

```
if (condition) code to be
    executed if  condition is true;
elseif (condition) code to be
    executed if  condition is true; ?>
else code to be executed if
    condition is false;
```

```
<?php
$num=12;
if($num%2==0){
echo "$num is even number";
}else{
echo "$num is odd number";
}
?>
```

PHP Switch Statement

- If you want to select one of many blocks of code to be executed, use the Switch statement.
- The switch statement is used to avoid long blocks of if..elseif..else code.

Syntax

switch (*expression*)
{
case *label1: code to be executed if expression = label1;*
break;
case *label2: code to be executed if expression = label2;*
break;
default: *code to be executed if expression is different from both label1 and label2;*
}

```
switch ($textcolor)
{
case "black":
    echo "I'm black";
    break;
case "blue":
    echo "I'm blue";
    break;
case "red":
    echo "I'm red";
    break;
default:  // It must be
    something else
    echo "too bad!!, I'm
    something else";
}
```

PHP Looping

- Looping statements in PHP are used to execute the same block of code a specified number of times.
- In PHP we have the following looping statements:
  - while - loops through a block of code if and as long as a specified condition is true
  - do...while - loops through a block of code once, and then repeats the loop as long as a special condition is true
  - for - loops through a block of code a specified number of times
  - foreach - loops through a block of code for each element in an array

The while Statement

Syntax

```
while (condition)
    {
    / /  statements
    }
```

Example

```
    <html>  <head>  <title>Let us count !!!</title></head>
     <body>
    <?php
      $limit = 10;
      echo "<h2> Let us count from 1 to $limit</h2><br
       />";
      $count = 1;
      while ($count <= $limit)
      {
        echo "counting $count of $limit <br>";
        $count++;
      }
    ?>
        </body>
```

The do...while Statement

- The do...while statement will execute a block of code at least once - it then will repeat the loop as long as a condition is true.

Syntax

- do { *code to be executed;* } while (*condition*);

Example

```
<html> <body>
<?php
$i=0;
do { $i++; echo "The number
is " . $i . "<br />"; }
while ($i<5);
?>
</body> </html>
```

The for Statement

- It is used when you know how many times you want to execute a statement or a list of statements.

Syntax

- for (*init*; *cond*; *incr*) { *code to be executed;* } Parameters:

- **init**: Is mostly used to set a counter, but can be any code to be executed once at the beginning of the loop statement.
- **cond**: Is evaluated at beginning of each loop iteration. If the condition evaluates to TRUE, the loop continues and the code executes. If it evaluates to FALSE, the execution of the loop ends.
- **incr**: Is mostly used to increment a counter, but can be any code to be executed at the end of each loop.

Example

```
<html> <body>
<?php
for ($i=1; $i<=5; $i++)
{
echo "Hello World!<br
/>";
}
?>
</body> </html>
```

The foreach Statement

- The foreach statement is used to loop through arrays.
- The foreach loop works only on arrays, and is used to loop through each key/value pair in an array
- For every loop, the value of the current array element is assigned to $value (and the array pointer is moved by one)
- - so on the next loop, you'll be looking at the next element.

Syntax:

```
foreach ($array as $value) {
        code to be executed;
}
```

- For every loop iteration, the value of the current array element is assigned to $value and the array pointer is moved by one, until it reaches the last array element.
- The foreach Statement

The foreach Statement

Example

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

Out Put

```
red
green
blue
yellow
```

The foreach Statement

Example

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

Out Put

```
red
green
blue
yellow
```

Arrays

- Arrays are complex variables that allow us to store more than one value or a group of values under a single variable name.
- There are three types of arrays that you can create. These are:
- Indexed array — An array with a numeric key.
- Associative array — An array where each key has its own specific value.
- Multidimensional array — An array containing one or more arrays within itself.

Indexed Arrays

- An indexed or numeric array stores each array element with a numeric index. The following examples shows two ways of creating an indexed array.
- `<?php`
- `// Define an indexed array`
- `$colors = array("Red", "Green", "Blue"); ?>`
- In an indexed or numeric array, the indexes are automatically assigned and start with 0, and the values can be any data type.

Indexed Arrays

```
<html>
   <body>
         <?php
         /* First method to create array. */
         $numbers = array( 1, 2, 3, 4, 5);
        foreach( $numbers as $value ) {
            echo "Value is $value <br />";
         }
?>
    </body>
</html>
```

Output :

```
 Value is 1
Value is 2
Value is 3
Value is 4
Value is 5
```

Indexed Arrays

```
<html>
   <body>
         <?php
/* Second method to create array. */
         $numbers[0] = "one";
         $numbers[1] = "two";
         $numbers[2] = "three";
         $numbers[3] = "four";
         $numbers[4] = "five";
            foreach( $numbers as $value ) {
            echo "Value is $value <br />";
         }
      ?> </body>
</html>
```

Output:

```
Value is one
Value is two
Value is three
Value is four
Value is five
```

Associative array

- The associative arrays are very similar to numeric arrays in term of functionality but they are different in terms of their index.
- Associative array will have their index as string so that you can establish a strong association between key and values.
- To store the salaries of employees in an array, a numerically indexed array would not be the best choice. Instead, we could use the employees names as the keys in our associative array, and the value would be their respective salary.
- Don't keep associative array inside double quote while printing otherwise it would not return any value.

Associative array

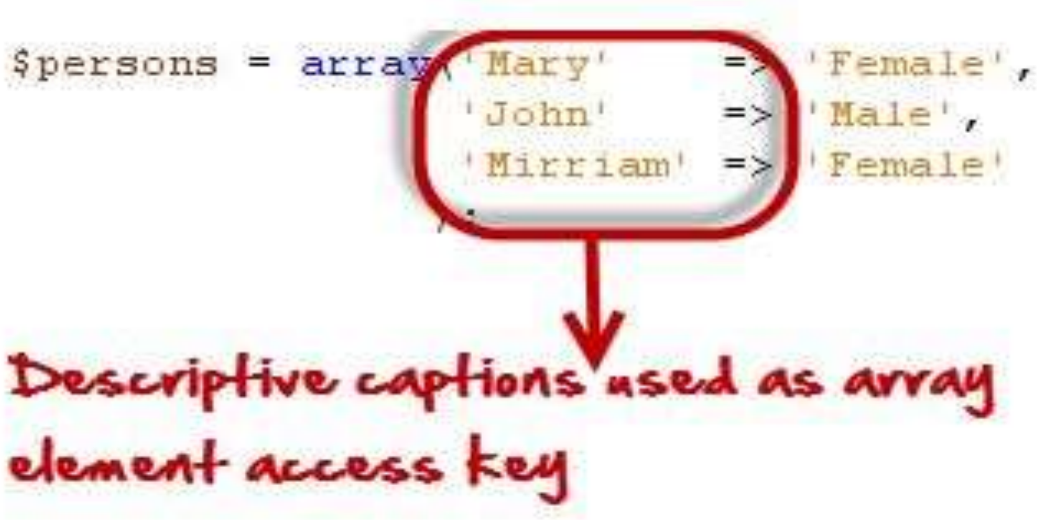
- Syntax:
- <?php
- $variable_name['key_name’]=value;
- Or
- $variable_name = array('keyname' => value); ?>

Associative array

- “$variable_name…” is the name of the variable
- “['key_name']” is the access index number of the element
- “value” is the value assigned to the array element.
- <?php
- $persons = array(‘Mary’ => "Female", "John" => "Male", "Mirriam" => "Female");
- print_r($persons);
- echo "";
- echo "Mary is a " . $persons["Mary"];
- ?>

Associative array

- Output:
- Array ([Mary] => Female [John] => Male [Mirriam] => Female)
- Mary is a Female
- Associative array are also very useful when retrieving data from the database.



```
$persons = array('Mary'    => 'Female',
                 'John'    => 'Male',
                 'Mirriam' => 'Female'
);
```

Multidimensional Array

- A multidimensional array is an array that contains one or more arrays as its elements.
- Each array element in a multidimensional array can also be an array, forming a hierarchy of arrays.
- This can be useful when working with complex data structures such as tables, matrices, or nested lists.
- In PHP, you can create a multidimensional array by defining an array with square brackets and nesting other arrays inside it.

Multidimensional Array

- For example, the following code creates a two-dimensional array that contains the names and ages of three people:
- $people = array(
-     array("Alice", 25),
-     array("Bob", 30),
-     array("Charlie", 20)
-);
- Here, $people is a two-dimensional array with three sub-arrays, each representing a person. Each sub-array has two elements, the first element being the name of the person and the second element being their age.

Multidimensional Array

- To access an element of a multidimensional array, you need to specify the index for each level of the array. For example, to access the age of the second person in the array, you would use the following code:
- echo $people[1][1]; // Output: 30
- Here, $people[1] refers to the second sub-array, which contains the name and age of Bob.
- $people[1][1] refers to the second element of this sub-array, which is Bob's age.

Multidimensional Array

- You can also loop through a multidimensional array using nested loops. For example, the following code loops through the $people array and displays each person's name and age:
- foreach ($people as $person) {
-     echo "Name: " . $person[0] . ", Age: " . $person[1] . "<br>";
- } This will output:
- Name: Alice, Age: 25
- Name: Bob, Age: 30
- Name: Charlie, Age: 20

Array Functions

1. **count**(): This function is used to count the number of elements in an array.

- It takes one argument, which is the array to be evaluated, and returns the number of elements in that array. For example:
- $fruits = array("apple", "banana", "cherry");
- echo count($fruits); // Output: 3

Array Functions

2. **array_push():** This function is used to add one or more elements to the end of an array.

- It takes two or more arguments: the original array and the element(s) to be added to the end of that array. For example:
- $fruits = array("apple", "banana", "cherry");
- array_push($fruits, "orange", "kiwi");
- print_r($fruits); // Output: Array ([0] => apple [1] => banana [2] => cherry [3] => orange [4] => kiwi)

Array Functions

3. **array_pop**(): This function is used to remove the last element from an array.

- It takes one argument, which is the array from which the last element is to be removed. For example:
- $fruits = array("apple", "banana", "cherry");
- array_pop($fruits);
- print_r($fruits); // Output: Array ([0] => apple [1] => banana)

Array Functions

4) **array_merge():** This function is used to merge two or more arrays into a single array.

- The syntax for using the array_merge() function is array_merge($array1, $array2, ...).
- Here, $array1, $array2, etc. are the arrays that need to be merged.
- $fruits1 = array("apple", "banana", "cherry");
- $fruits2 = array("orange", "kiwi");
- $fruits = array_merge($fruits1, $fruits2);
- print_r($fruits); // Output: Array ([0] => apple [1] => banana [2] => cherry [3] => orange [4] => kiwi)

Array Functions

5) **array_search()**: This function is used to search a given array for a specific value and return the corresponding key if found.

- The syntax for using the array_search() function is array_search($value, $array). Here, $value is the value that needs to be searched and $array is the array in which the search needs to be performed.

```
$fruits = array("apple", "banana", "cherry");
$key = array_search("banana", $fruits);
echo $key; // Output: 1
```