**PHP Notes**

1. Comments :-

// This is a single-line comment

# This is also a single-line comment

/\* This is a

multi-line comment \*/

1. Variables :-

* A variable starts with the $ 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
* Remember that PHP variable names are case-sensitive!
* PHP has no command for declaring a variable, and the data type depends on the value of the variable.
* PHP supports the following data types:
* String
* Integer
* Float (floating point numbers - also called double)
* Boolean
* Array
* Object
* NULL
* Resource
* To get the data type of a variable, use the var\_dump() function.
* PHP Variables Scope

PHP has three different variable scopes:

* Local :- A variable declared **within** a function has a LOCAL SCOPE and can only be accessed within that function
* Global :- A variable declared **outside** a function has a GLOBAL SCOPE and can only be accessed outside a function
  + The global keyword is used to access a global variable from within a function.
  + PHP also stores all global variables in an array called $GLOBALS[*index*]. The *index* holds the name of the variable.
* Static :- when a function is completed/executed, all of its variables are deleted. However, sometimes we want a local variable NOT to be deleted. We need it for a further job.

3)Array:-

Add array item

Ex:-

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

$fruits[] = "Orange";

Add in Associative array

Ex:-

$cars = array("brand" => "Ford", "model" => "Mustang");

$cars["color"] = "Red";

Add Multiple Array Items

Ex:-

$fruits = array(“Apple”, ”Banana”, “Orange”);

array\_push($fruits, “Kiwi”, “Lemon”);

Remove array item

Ex:-

$fruits = array(“Apple”, ”Banana”, “Orange”);

array\_splice($fruits, 1,1);

array\_splice($fruits, 1, 2); // for multiple items removal

Using the unset Function

Ex:- $cars = array("Volvo", "BMW", "Toyota");

unset($cars[1]);

unset($cars[0], $cars[1]); // for ,multiple items removal

Remove Item From an Associative Array

Ex:- $cars = array("brand" => "Ford", "model" => "Mustang", "year" => 1964);

unset($cars["model"]);

Using array\_diff function

This function returns a new array, without the specified items.

$cars = array("brand" => "Ford", "model" => "Mustang", "year" => 1964);

$newarray = array\_diff($cars, ["Mustang", 1964]);

Sort Functions for array:-

* 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

Array functions :-

1] array()

2] array\_change\_key\_case(*array*,*case*) //CASE\_UPPER or CASE\_LOWER

3] array\_chunk(*array*,*size*,*preserve\_key*)

Split an array into chunks of two:

$cars=array("Volvo","BMW","Toyota","Honda","Mercedes","Opel");  
print\_r(array\_chunk($cars,2));

4] array\_column(*array*,*column\_key*,*index\_key*)

5] array\_combine(*keys*,*values*)

6] array\_count\_values(*array*)

The array\_count\_values() function counts all the values of an array

7] array\_diff(*array1, array2, array3, ...*)

8] array\_fill(index, number, value)

9] array\_filter(array, callbackfunction, flag)

10] array\_flip(*array*)

11] Compare the**values** of two arrays, and return the matches

array\_intersect(array1, array2, array3, …)  
12] array\_key\_exists(*key, array*)

13] array\_keys(array)

14] array\_map(myfunction, array1, array2, array3, …)

15] array\_merge(*array1, array2, array3, ...*) // normal two arrays

16] array\_merge\_recursive(*array1, array2, array3, ...*) //two associative arrays. And if anything is common then it will also be stored as array in new array.

17] array\_multisort(*array1, sortorder, sorttype, array2, array3, ...*)

It sorts the array1 and adjust the other arrays in correspondence

18] array\_reduce(*array, myfunction, initial*)

19] array\_shift(*array*)

The array\_shift() function removes the first element from an array, and returns the value of the removed element.

20] array\_slice(*array, start, length, preserve*)

4) Form validation

The $\_SERVER["PHP\_SELF"] is a super global variable that returns the filename of the currently executing script.

The htmlspecialchars() function converts special characters into HTML entities. This means that it will replace HTML characters like < and > with &lt; and &gt;

we check whether the form has been submitted using $\_SERVER["REQUEST\_METHOD"]. If the REQUEST\_METHOD is POST, then the form has been submitted - and it should be validated. If it has not been submitted, skip the validation and display a blank form.

5) Php Advanced

i) Date and time

echo "Today is " . date("Y/m/d") . "<br>";  
echo "Today is " . date("Y.m.d") . "<br>";  
echo "Today is " . date("Y-m-d") . "<br>";

* 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)

date\_default\_timezone\_set("America/New\_York");

$d=strtotime("10:30pm April 15 2014");  
echo "Created date is " . date("Y-m-d h:i:sa", $d);

ii) Include

there is one big difference between include and require; when a file is included with the include statement and PHP cannot find it, the script will continue to execute

iii) File handling

- The readfile() function reads a file and writes it to the output buffer.

echo readfile("webdictionary.txt");

- The fgets() function is used to read a single line from a file.

echo fgets($myfile);

-The feof() function checks if the "end-of-file" (EOF) has been reached. The feof() function is useful for looping through data of unknown length.

while(!feof($myfile)) {  
  echo fgets($myfile) . "<br>";  
}

-The fopen() function is also used to create a file. Maybe a little confusing, but in PHP, a file is created using the same function used to open files.

$myfile = fopen("testfile.txt", "w")

-You can append data to a file by using the "a" mode. The "a" mode appends text to the end of the file, while the "w" mode overrides (and erases) the old content of the file.

$myfile = fopen("newfile.txt", "a") or die("Unable to open file!");  
$txt = "Donald Duck\n";  
fwrite($myfile, $txt);

iv) File upload

-The form sends the file to a PHP script using POST method with enctype="multipart/form-data".

-PHP uses the $\_FILES superglobal to **receive** and **process** the file.

-You can **move the uploaded file** to a folder using move\_uploaded\_file().

-Code for it , we can see in w3school.

* $target\_dir = "uploads/" - specifies the directory where the file is going to be placed
* $target\_file specifies the path of the file to be uploaded
* $uploadOk=1 is not used yet (will be used later)
* $imageFileType holds the file extension of the file (in lower case)

v) Cookies

The following example creates a cookie named "user" with the value "John Doe". The cookie will expire after 30 days (86400 \* 30). The "/" means that the cookie is available in entire website (otherwise, select the directory you prefer).

We then retrieve the value of the cookie "user" (using the global variable $\_COOKIE). We also use the isset() function to find out if the cookie is set

<?php  
$cookie\_name = "user";  
$cookie\_value = "John Doe";  
setcookie($cookie\_name, $cookie\_value, time() + (86400 \* 30), "/"); // 86400 = 1 day  
?>

<html>  
<body>

<?php  
if(!isset($\_COOKIE[$cookie\_name])) {  
  echo "Cookie named '" . $cookie\_name . "' is not set!";  
} else {  
  echo "Cookie '" . $cookie\_name . "' is set!<br>";  
  echo "Value is: " . $\_COOKIE[$cookie\_name];  
}  
?>

</body>  
</html>

vi) Sessions :

A session is started with the session\_start() function. Session variables are set with the PHP global variable: $\_SESSION. Now, let's create a new page called "demo\_session1.php". In this page, we start a new PHP session and set some session variables

<?php  
// Start the session  
session\_start();  
?>  
<!DOCTYPE html>  
<html>  
<body>  
<?php  
// Set session variables  
$\_SESSION["favcolor"] = "green";

$\_SESSION["favanimal"] = "cat";  
echo "Session variables are set.";  
?>  
</body>  
</html>

-Get PHP Session Variable Values

echo "Favorite color is " . $\_SESSION["favcolor"] . ".<br>";  
echo "Favorite animal is " . $\_SESSION["favanimal"] . ".";

-Another way to show all the session variable values for a user session is to run the following code.

print\_r($\_SESSION);

-Modify a PHP Session Variable

$\_SESSION["favcolor"] = "yellow";

-Destroy a PHP Session

// remove all session variables  
session\_unset();  
  
// destroy the session  
session\_destroy();

vii) Filter

PHP filters are used to validate and sanitize external input.

-Sanitizing a String : filter\_var() function to remove all HTML tags from a string

$str = "<h1>Hello World!</h1>";  
$newstr = filter\_var($str, FILTER\_SANITIZE\_STRING);  
echo $newstr;

-he following example uses the filter\_var() function to check if the variable $int is an integer. If $int is an integer, the output of the code below will be: "Integer is valid". If $int is not an integer, the output will be: "Integer is not valid"

<?php  
$int = 100;  
  
if (!filter\_var($int, FILTER\_VALIDATE\_INT) === false) {  
  echo("Integer is valid");  
} else {  
  echo("Integer is not valid");  
}  
?>

-validate an ip

$ip = "127.0.0.1";  
  
if (!filter\_var($ip, FILTER\_VALIDATE\_IP) === false) {  
  echo("$ip is a valid IP address");  
} else {  
  echo("$ip is not a valid IP address");  
}

- Sanitize and Validate an Email Address

$email = "john.doe@example.com";  
  
// Remove all illegal characters from email  
$email = filter\_var($email, FILTER\_SANITIZE\_EMAIL);  
  
// Validate e-mail  
if (!filter\_var($email, FILTER\_VALIDATE\_EMAIL) === false) {  
  echo("$email is a valid email address");  
} else {  
  echo("$email is not a valid email address");  
}

ix) Php callback function

A callback function is a function which is passed as an argument into another function.

-Run a callback from a user-defined function

<?php  
function exclaim($str) {  
  return $str . "! ";  
}  
  
function ask($str) {  
  return $str . "? ";  
}  
  
function printFormatted($str, $format) {  
  // Calling the $format callback function  
  echo $format($str);  
}

// Pass "exclaim" and "ask" as callback functions to printFormatted()  
printFormatted("Hello world", "exclaim");  
printFormatted("Hello world", "ask");  
?>

x) Php Json : JSON stands for JavaScript Object Notation, and is a syntax for storing and exchanging data.

PHP has some built-in functions to handle JSON.

First, we will look at the following two functions:

* json\_encode()

$cars = array("Volvo", "BMW", "Toyota");  
  
echo json\_encode($cars);

* json\_decode()

$jsonobj = '{"Peter":35,"Ben":37,"Joe":43}';  
  
var\_dump(json\_decode($jsonobj));

The json\_decode() function returns an object by default. The json\_decode() function has a second parameter, and when set to true, JSON objects are decoded into associative arrays.

$obj = json\_decode($jsonobj);

$arr = json\_decode($jsonobj, true);

xi) Exception : An exception is an object that describes an error or unexpected behaviour of a PHP script.

Exceptions are thrown by many PHP functions and classes.

User defined functions and classes can also throw exceptions.

Exceptions are a good way to stop a function when it comes across data that it cannot use.

- The throw statement allows a user defined function or method to throw an exception. When an exception is thrown, the code following it will not be executed.

function divide($dividend, $divisor) {  
  if($divisor == 0) {  
    throw new Exception("Division by zero");  
  }  
  return $dividend / $divisor;  
}

-The try catch statement

To avoid the error from the example above, we can use the try...catch statement to catch exceptions and continue the process.

Code in the finally block will always run regardless of whether an exception was caught.

try {  
  echo divide(5, 0);  
} catch(Exception $e) {  
  echo "Unable to divide.";  
} finally {  
  echo "Process complete.";  
}

The Exception Object contains information about the error or unexpected behaviour that the function encountered.

|  |  |
| --- | --- |
| getMessage() | Returns a string describing why the exception was thrown |
| getPrevious() | If this exception was triggered by another one, this method returns the previous exception. If not, then it returns *null* |
| getCode() | Returns the exception code |
| getFile() | Returns the full path of the file in which the exception was thrown |
| getLine() | Returns the line number of the line of code which threw the exception |

When catching an exception, the following table shows some of the methods that can be used to get information about the exception

6) OOP

There are two ways to change value of the property.

1

class Fruit {  
  public $name;  
  function set\_name($name) {  
    $this->name = $name;  
  }  
}  
$apple = new Fruit();  
$apple->set\_name("Apple");

2

class Fruit {  
  public $name;  
}  
$apple = new Fruit();  
$apple->name = "Apple";

You can use the instanceof keyword to check if an object belongs to a specific class

<?php  
$apple = new Fruit();  
var\_dump($apple instanceof Fruit);  
?>

- \_\_Constructor

A constructor allows you to initialize an object's properties upon creation of the object.

function \_\_construct($name) {  
    $this->name = $name;  
  }

-\_\_Destructor

A destructor is called when the object is destructed or the script is stopped or exited.

  function \_\_destruct() {  
    echo "The fruit is {$this->name}.";  
  }

- Access Modifiers

* public - the property or method can be accessed from everywhere. This is default
* protected - the property or method can be accessed within the class and by classes derived from that class
* private - the property or method can ONLY be accessed within the class

<?php  
class Fruit {  
  public $name;  
  protected $color;  
  private $weight;  
}  
  
$mango = new Fruit();  
$mango->name = 'Mango'; // OK  
$mango->color = 'Yellow'; // ERROR  
$mango->weight = '300'; // ERROR  
?>

-same for the methods too

-Inheritance

The child class will inherit all the public and protected properties and methods from the parent class. In addition, it can have its own properties and methods.

// Strawberry is inherited from Fruit  
class Strawberry extends Fruit {  
  public function message() {  
    echo "Am I a fruit or a berry? ";  
  }  
}

The final keyword can be used to prevent class inheritance or to prevent method overriding.

-Constants

Class constants can be useful if you need to define some constant data within a class.

A class constant is declared inside a class with the const keyword.

A constant cannot be changed once it is declared.

Class constants are case-sensitive. However, it is recommended to name the constants in all uppercase letters.

We can access a constant from outside the class by using the class name followed by the scope resolution operator (::) followed by the constant name.

<?php  
class Goodbye {  
  const LEAVING\_MESSAGE = "Thank you for visiting W3Schools.com!";  
}  
  
echo Goodbye::LEAVING\_MESSAGE;  
?>

-Abstract

Abstract classes and methods are when the parent class has a named method, but need its child class(es) to fill out the tasks.

An abstract class is a class that contains at least one abstract method. An abstract method is a method that is declared, but not implemented in the code.

-Static method and properties

A **static property** or **method** belongs to the **class itself**, not to any object.

class Calculator {

public static $pi = 3.14159;

public static function square($n) {

return $n \* $n;

}

}

// Accessing without creating an object

echo Calculator::$pi . "<br>"; // ✅ 3.14159

echo Calculator::square(5); // ✅ 25

| **Location** | **How to Access** |
| --- | --- |
| Outside class | ClassName::$property or ClassName::method() |
| Inside class | self::$property or self::method() |
| In child class | parent::method() if inherited |