UNIT 3 (INTERNET PROGRAMMING/ PHP)

- Arrays: Array – Key pair values
- Array functions, is SET, UNSET, gettype(), settype(),
- Control statements (if, switch),
- Loops,
- User Defined Functions (with argument, return values),
- Global variable
- default value
- GET - POST method
- URL encoding
- HTML Encoding
- Cookies, Sessions
- Include statement.
- File:read and write from the file.
- Ethical use of features of PHP.

Array:

An array is a data structure that stores one or more similar type of values in a single value. For example if you want to store 100 numbers then instead of defining 100 variables its easy to define an array of 100 length.

There are three different kind of arrays and each array value is accessed using an ID c which is called array index.

- **Numeric array −** An array with a numeric index. Values are stored and accessed in linear fashion.
- **Associative array −** An array with strings as index. This stores element values in association with key values rather than in a strict linear index order.
- **Multidimensional array −** An array containing one or more arrays and values are accessed using multiple indices

NOTE − Built-in array functions is given in function reference PHP Array Functions

Numeric Array

These arrays can store numbers, strings and any object but their index will be represented by numbers. By default array index starts from zero.

Example

Following is the example showing how to create and access numeric arrays.

Here we have used **array()** function to create array. This function is explained in function reference.

```
<html>
  <body>

     <?php
        /* First method to create array. */
        $numbers = array( 1, 2, 3, 4, 5);

        foreach( $numbers as $value ) {
           echo "Value is $value <br />";
        }

        /* 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>
```

This will produce the following result −

```
Value is 1
Value is 2
Value is 3
Value is 4
Value is 5
Value is one
Value is two
Value is three
Value is four
Value is five
```

Associative Arrays/ Array – Key pair values:

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.

NOTE − Don't keep associative array inside double quote while printing otherwise it would not return any value.

Example

```
<html>
   <body>

      <?php
         /* First method to associate create array. */
         $salaries = array("mohammad" => 2000, "qadir" => 1000, "zara"
=> 500);

         echo "Salary of mohammad is ". $salaries['mohammad'] . "<br />";
         echo "Salary of qadir is ".  $salaries['qadir']. "<br />";
         echo "Salary of zara is ".  $salaries['zara']. "<br />";

         /* Second method to create array. */
         $salaries['mohammad'] = "high";
         $salaries['qadir'] = "medium";
         $salaries['zara'] = "low";
         
         echo "Salary of mohammad is ". $salaries['mohammad'] . "<br />";
         echo "Salary of qadir is ".  $salaries['qadir']. "<br />";
         echo "Salary of zara is ".  $salaries['zara']. "<br />";
      ?>
   
   </body>
</html>
```

This will produce the following result −

```
Salary of mohammad is 2000
Salary of qadir is 1000
Salary of zara is 500
Salary of mohammad is high
Salary of qadir is medium
Salary of zara is low
```

Multidimensional Arrays

A multi-dimensional 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. Values in the multi-dimensional array are accessed using multiple index.

Example

In this example we create a two dimensional array to store marks of three students in three subjects −

This example is an associative array, you can create numeric array in the same fashion.

```
<html>
   <body>
      
      <?php
         $marks = array( 
            "mohammad" => array (
               "physics" => 35,
               "maths" => 30,
               "chemistry" => 39
            ),

            "qadir" => array (
               "physics" => 30,
               "maths" => 32,
               "chemistry" => 29
            ),

            "zara" => array (
               "physics" => 31,
               "maths" => 22,
               "chemistry" => 39
            )
         );

         /* Accessing multi-dimensional array values */
         echo "Marks for mohammad in physics : " ;
         echo $marks['mohammad']['physics'] . "<br />";

         echo "Marks for qadir in maths : ";
         echo $marks['qadir']['maths'] . "<br />";

         echo "Marks for zara in chemistry : " ;
         echo $marks['zara']['chemistry'] . "<br />";
      ?>

   </body>
</html>
```

This will produce the following result −

```
Marks for mohammad in physics : 35
Marks for qadir in maths : 32
Marks for zara in chemistry : 39
```

PHP | settype() Function

The settype() function is a built-in function in PHP. The settype() function is used to the set the type of a variable. It is used to set type or modify type of an existing variable.

Syntax:

boolean settype($variable_name, $type)

Parameters: The settype() function accepts two parameters as shown in above syntax and are described below.

$variable_name: This parameter specifies the name of variable whose type we want to modify. This parameter can be of any type that is, it can be of integer type or a string type etc.

$type: This parameter specifies the type of variable that is needed. Possible values of this parameter are: boolean, integer, float, string, array, object, null.

Return value: This function returns a boolean type value. It returns TRUE in case of success and FALSE in case of failure.

Below programs illustrate the settype() function in PHP:

Program 1:

```
<?php

// PHP program to illustrate settype() function

$var1 = "123xyz";

$var2 = 3;

$r = true;

settype($var1, "integer");

settype($var2, "float");

settype($r, "string");

echo $var1."\n";

echo $var2."\n";
echo $r."\n";

?>
```

Output:

```
123

3

1
```

PHP | gettype() Function

The gettype() function is an inbuilt function in PHP which is used to get the type of a variable. It is used to check the type of existing variable.

Syntax:

```
string gettype ( $var )
```

Parameter: This function accepts a single parameter $var. It is the name of variable which is needed to be checked for type of variable.

Return Value: This function returns a string type value. Possible values for the returned string are:

boolean
integer
double (for historical reasons “double” is returned in case of float)
string
array
object
resource
NULL
unknown type

Below programs illustrate the gettype() function in PHP:

Program 1:

```php
<?php
// PHP program to illustrate gettype() function
$var1 = true; // boolean value
$var2 = 3; // integer value
$var3 = 5.6; // double value
$var4 = "Abc3462"; // string value
$var5 = array(1, 2, 3); // array value
$var6 = new stdClass; // object value
$var7 = NULL; // null value
$var8 = tmpfile(); // resource value

echo gettype($var1)."\n";
echo gettype($var2)."\n";
echo gettype($var3)."\n";
echo gettype($var4)."\n";
echo gettype($var5)."\n";
echo gettype($var6)."\n";
echo gettype($var7)."\n";
echo gettype($var8)."\n";
?>
```

Output:

```
boolean
```

integer
double
string
array
object
NULL
resource

Array UNSET() Method :

unset(mixed $var, mixed ...$vars): void

- unset() destroys the specified variables.
- The behavior of unset() inside of a function can vary depending on what type of variable you are attempting to destroy.
- If a globalized variable is unset() inside of a function, only the local variable is destroyed. The variable in the calling environment will retain the same value as before unset() was called.
- To unset() a global variable inside of a function, then use the *$GLOBALS* array to do so
- If a variable that is PASSED BY REFERENCE is unset() inside of a function, only the local variable is destroyed. The variable in the calling environment will retain the same value as before unset() was called.
- If a variable that is PASSED BY REFERENCE is unset() inside of a function, only the local variable is destroyed. The variable in the calling environment will retain the same value as before unset() was called.

Parameters ¶

var The variable to be unset.

vars Further variables.

Array Functions:

Table of Contents:

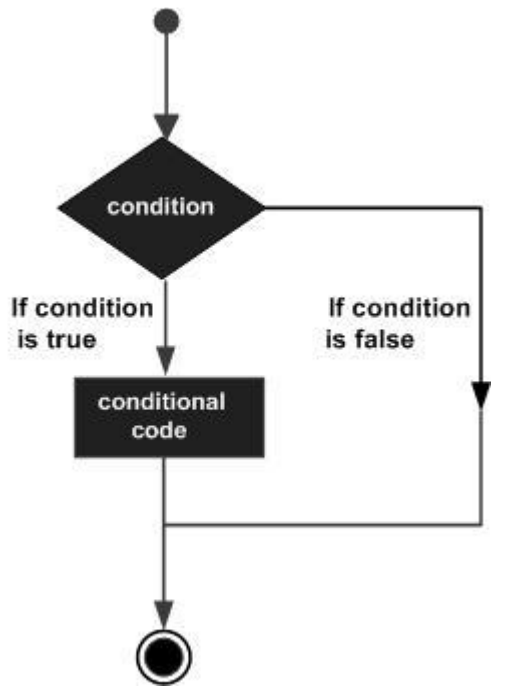
- array_map — Applies the callback to the elements of the given arrays
- array_merge_recursive — Merge one or more arrays recursively
- array_merge — Merge one or more arrays
- array_multisort — Sort multiple or multi-dimensional arrays
- array_pad — Pad array to the specified length with a value
- array_pop — Pop the element off the end of array
- array_product — Calculate the product of values in an array
- array_push — Push one or more elements onto the end of array
- array_rand — Pick one or more random keys out of an array
- array_reduce — Iteratively reduce the array to a single value using a callback function
- array_replace_recursive — Replaces elements from passed arrays into the first array recursively
- array_replace — Replaces elements from passed arrays into the first array
- array_reverse — Return an array with elements in reverse order
- array_search — Searches the array for a given value and returns the first corresponding key if successful
- array_shift — Shift an element off the beginning of array
- array_slice — Extract a slice of the array
- array_splice — Remove a portion of the array and replace it with something else
- array_sum — Calculate the sum of values in an array
- array_udiff_assoc — Computes the difference of arrays with additional index check, compares data by a callback function
- array_udiff_uassoc — Computes the difference of arrays with additional index check, compares data and indexes by a callback function
- array_udiff — Computes the difference of arrays by using a callback function for data comparison
- array_uintersect_assoc — Computes the intersection of arrays with additional index check, compares data by a callback function
- array_uintersect_uassoc — Computes the intersection of arrays with additional index check, compares data and indexes by separate callback functions
- array_uintersect — Computes the intersection of arrays, compares data by a callback function
- array_unique — Removes duplicate values from an array
- array_unshift — Prepend one or more elements to the beginning of an array
- array_values — Return all the values of an array

- array_walk_recursive — Apply a user function recursively to every member of an array
- array_walk — Apply a user supplied function to every member of an array
- array — Create an array
- arsort — Sort an array in descending order and maintain index association
- asort — Sort an array in ascending order and maintain index association
- compact — Create array containing variables and their values
- count — Counts all elements in an array or in a Countable object
- current — Return the current element in an array
- each — Return the current key and value pair from an array and advance the array cursor
- end — Set the internal pointer of an array to its last element
- extract — Import variables into the current symbol table from an array
- in_array — Checks if a value exists in an array
- key_exists — Alias of array_key_exists
- key — Fetch a key from an array
- krsort — Sort an array by key in descending order
- ksort — Sort an array by key in ascending order
- list — Assign variables as if they were an array
- natcasesort — Sort an array using a case insensitive "natural order" algorithm
- natsort — Sort an array using a "natural order" algorithm
- next — Advance the internal pointer of an array
- pos — Alias of current
- prev — Rewind the internal array pointer
- range — Create an array containing a range of elements
- reset — Set the internal pointer of an array to its first element
- rsort — Sort an array in descending order
- shuffle — Shuffle an array
- sizeof — Alias of count
- sort — Sort an array in ascending order
- uasort — Sort an array with a user-defined comparison function and maintain index association
- uksort — Sort an array by keys using a user-defined comparison function
- usort — Sort an array by values using a user-defined comparison function

PHP - Decision Making

The if, elseif ...else and switch statements are used to take decision based on the different condition.

You can use conditional statements in your code to make your decisions. PHP supports following three decision making statements −



- **if...else statement** − use this statement if you want to execute a set of code when a condition is true and another if the condition is not true
- **elseif statement** − is used with the if...else statement to execute a set of code if **one** of the several condition is true
- **switch statement** − is used 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.

The If...Else Statement

If you want to execute some code if a condition is true and another code if a condition is false, use the if....else statement.

Syntax

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

Example

The following example will output "Have a nice weekend!" if the current day is Friday, Otherwise, it will output "Have a nice day!":

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

It will produce the following result −

```
Have a nice weekend!
```

The ElseIf Statement

If you want to execute some code if one of the several conditions are 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;
```

Example

The following example will output "Have a nice weekend!" if the current day is Friday, and "Have a nice Sunday!" if the current day is Sunday. Otherwise, it will output "Have a nice day!" −

```
<html>
  <body>

    <?php
      $d = date("D");

      if ($d == "Fri")
        echo "Have a nice weekend!";

      elseif ($d == "Sun")
        echo "Have a nice Sunday!";

      else
        echo "Have a nice day!";
    ?>

  </body>
</html>
```

It will produce the following result −

```
Have a nice Weekend!
```

The 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;
}
```

PHP - Loop Types

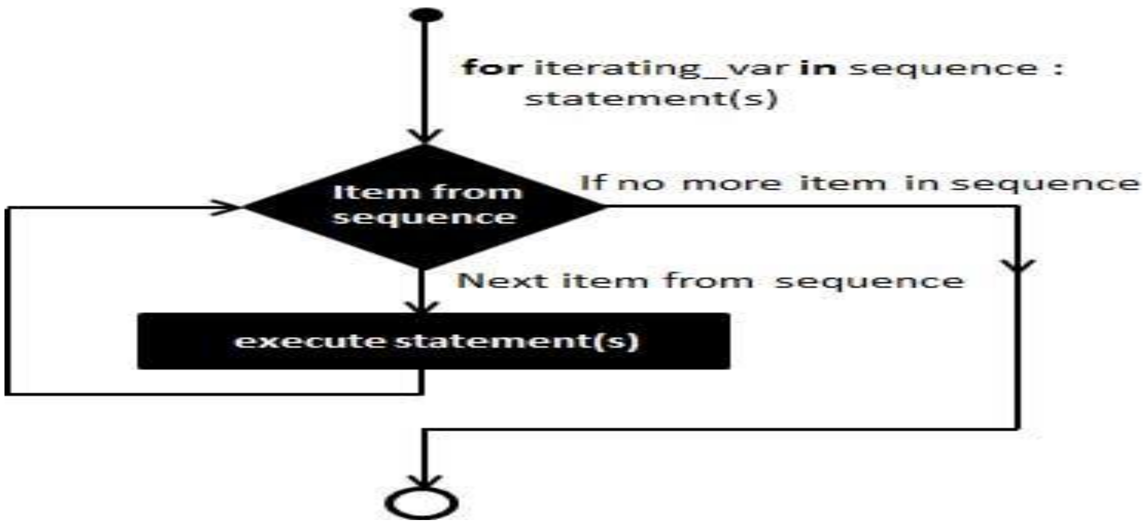
Loops in PHP are used to execute the same block of code a specified number of times. PHP supports following four loop types.

- **for** − loops through a block of code a specified number of times.
- **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.
- **foreach** − loops through a block of code for each element in an array.

We will discuss about **continue** and **break** keywords used to control the loops execution.

The for loop statement

The for statement is used when you know how many times you want to execute a statement or a block of statements.



Syntax

```
for (initialization; condition; increment){
   code to be executed;
}
```

The initializer is used to set the start value for the counter of the number of loop iterations. A variable may be declared here for this purpose and it is traditional to name it $i.

Example

The following example makes five iterations and changes the assigned value of two variables on each pass of the loop −

```
<html>
   <body>

      <?php
         $a = 0;
         $b = 0;

         for( $i = 0; $i<5; $i++ ) {
            $a += 10;
            $b += 5;
         }

         echo ("At the end of the loop a = $a and b = $b" );
      ?>

   </body>
```

```
</html>
```

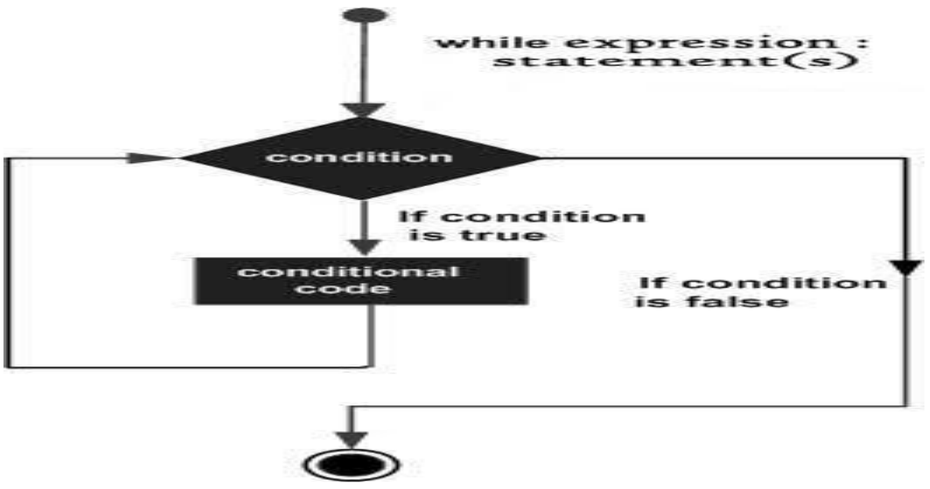
This will produce the following result −

```
At the end of the loop a = 50 and b = 25
```

The while loop statement

The while statement will execute a block of code if and as long as a test expression is true.

If the test expression is true then the code block will be executed. After the code has executed the test expression will again be evaluated and the loop will continue until the test expression is found to be false.



Syntax

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

Example

This example decrements a variable value on each iteration of the loop and the counter increments until it reaches 10 when the evaluation is false and the loop ends.

```
<html>
  <body>

    <?php
      $i = 0;
```

```
        $num = 50;

        while( $i < 10) {
           $num--;
           $i++;
        }

        echo ("Loop stopped at i = $i and num = $num" );
     ?>

   </body>
</html>
```

This will produce the following result −

```
Loop stopped at i = 10 and num = 40
```

The do...while loop 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

The following example will increment the value of i at least once, and it will continue incrementing the variable i as long as it has a value of less than 10 −

```
<html>
   <body>

     <?php
        $i = 0;
        $num = 0;

        do {
```

```
        $i++;
      }

      while( $i < 10 );
      echo ("Loop stopped at i = $i" );
    ?>

  </body>
</html>
```

This will produce the following result −

```
Loop stopped at i = 10
```

The foreach loop statement

The foreach statement is used to loop through arrays. For each pass the value of the current array element is assigned to $value and the array pointer is moved by one and in the next pass next element will be processed.

Syntax

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

Example

Try out following example to list out the values of an array.

```
<html>
  <body>

    <?php
      $array = array( 1, 2, 3, 4, 5);

      foreach( $array as $value ) {
        echo "Value is $value <br />";
      }
    ?>

  </body>
```

```
</html>
```

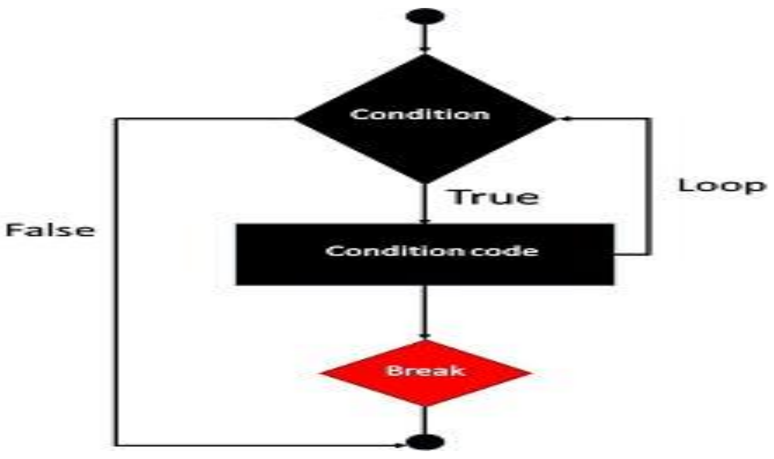
This will produce the following result −

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

The break statement

The PHP **break** keyword is used to terminate the execution of a loop prematurely.

The **break** statement is situated inside the statement block. It gives you full control and whenever you want to exit from the loop you can come out. After coming out of a loop immediate statement to the loop will be executed.



Example

In the following example condition test becomes true when the counter value reaches 3 and loop terminates.

```
<html>
  <body>

    <?php
      $i = 0;

      while( $i < 10) {
        $i++;
```

```
          if( $i == 3 )break;
        }
        echo ("Loop stopped at i = $i" );
      ?>

   </body>
</html>
```

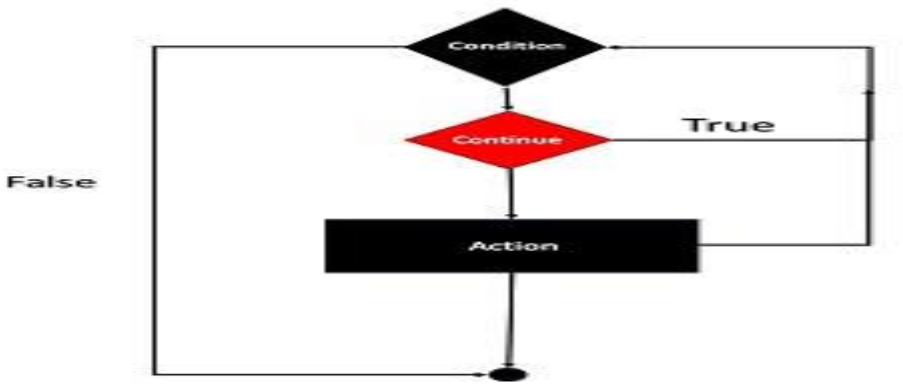
This will produce the following result −

```
Loop stopped at i = 3
```

The continue statement

The PHP **continue** keyword is used to halt the current iteration of a loop but it does not terminate the loop.

Just like the **break** statement the **continue** statement is situated inside the statement block containing the code that the loop executes, preceded by a conditional test. For the pass encountering **continue** statement, rest of the loop code is skipped and next pass starts.



Example

In the following example loop prints the value of array but for which condition becomes true it just skip the code and next value is printed.

```
<html>
   <body>

      <?php
        $array = array( 1, 2, 3, 4, 5);

        foreach( $array as $value ) {
          if( $value == 3 )continue;
          echo "Value is $value <br />";
        }
      ?>

   </body>
</html>
```

This will produce the following result −

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

PHP - Functions

PHP functions are similar to other programming languages. A function is a piece of code which takes one more input in the form of parameter and does some processing and returns a value.

There are two parts which should be clear to you −

- **Creating a PHP Function**
- **Calling a PHP Function**

Creating PHP Function

Its very easy to create your own PHP function. Suppose you want to create a PHP function which will simply write a simple message on your browser when you will call it. Following example creates a function called writeMessage() and then calls it just after creating it.

Note that while creating a function its name should start with keyword **function** and all the PHP code should be put inside { and } braces as shown in the following example below −

```
<html>

  <head>
    <title>Writing PHP Function</title>
  </head>

  <body>

    <?php
      /* Defining a PHP Function */
      function writeMessage() {
        echo "You are really a nice person, Have a nice time!";
      }

      /* Calling a PHP Function */
      writeMessage();
    ?>

  </body>
</html>
```

This will display following result −

```
You are really a nice person, Have a nice time!
```

PHP Functions with Parameters

PHP gives you option to pass your parameters inside a function. You can pass as many as parameters your like. These parameters work like variables inside your function. Following example takes two integer parameters and add them together and then print them.

```
<html>

  <head>
    <title>Writing PHP Function with Parameters</title>
  </head>

  <body>

    <?php
```

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

      addFunction(10, 20);
   ?>

  </body>
</html>
```

This will display following result −

```
Sum of the two numbers is : 30
```

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.

Following example depicts both the cases.

```
<html>

  <head>
    <title>Passing Argument by Reference</title>
  </head>

  <body>

    <?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 />";
      ?>

   </body>
</html>
```

This will display following result −

```
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.

```
<html>

   <head>
      <title>Writing PHP Function which returns value</title>
   </head>

   <body>

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

```
        $return_value = addFunction(10, 20);
         
        echo "Returned value from the function : $return_value";
     ?>
   
  </body>
</html>
```

This will display following result −

```
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.

```
<html>

  <head>
     <title>Writing PHP Function which returns value</title>
  </head>

  <body>

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

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

  </body>
</html>
```

This will produce following result −

```
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.

```
<html>

  <head>
    <title>Dynamic Function Calls</title>
  </head>

  <body>

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

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

  </body>
</html>
```

This will display following result −

```
Hello
```

PHP - GET & POST Methods

There are two ways the browser client can send information to the web server.

- **The GET Method**
- **The POST Method**

Before the browser sends the information, it encodes it using a scheme called URL encoding. In this scheme, name/value pairs are joined with equal signs and different pairs are separated by the ampersand.

```
name1=value1&name2=value2&name3=value3
```

Spaces are removed and replaced with the + character and any other nonalphanumeric characters are replaced with a hexadecimal values. After the information is encoded it is sent to the server.

The GET Method

The GET method sends the encoded user information appended to the page request. The page and the encoded information are separated by the **?** character.

http://www.test.com/index.htm?name1=value1&name2=value2

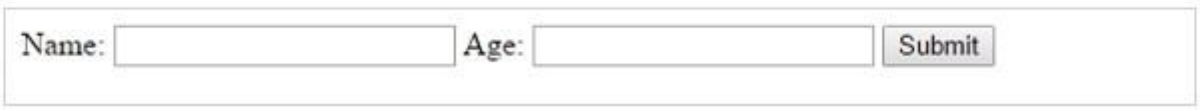
- The GET method produces a long string that appears in your server logs, in the browser's Location: box.
- The GET method is restricted to send upto 1024 characters only.
- Never use GET method if you have password or other sensitive information to be sent to the server.
- GET can't be used to send binary data, like images or word documents, to the server.
- The data sent by GET method can be accessed using QUERY_STRING environment variable.
- The PHP provides **$_GET** associative array to access all the sent information using GET method.

Try out following example by putting the source code in test.php script.

```
<?php
   if( $_GET["name"] || $_GET["age"] ) {
      echo "Welcome ". $_GET['name']. "<br />";
      echo "You are ". $_GET['age']. " years old.";

      exit();
   }
?>
<html>
   <body>

      <form action = "<?php $_PHP_SELF ?>" method = "GET">
         Name: <input type = "text" name = "name" />
         Age: <input type = "text" name = "age" />
         <input type = "submit" />
```

```
    </form>

  </body>
</html>
```

It will produce the following result −



The POST Method

The POST method transfers information via HTTP headers. The information is encoded as described in case of GET method and put into a header called QUERY_STRING.

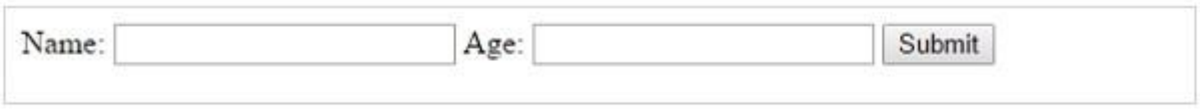
- The POST method does not have any restriction on data size to be sent.
- The POST method can be used to send ASCII as well as binary data.
- The data sent by POST method goes through HTTP header so security depends on HTTP protocol. By using Secure HTTP you can make sure that your information is secure.
- The PHP provides **$_POST** associative array to access all the sent information using POST method.

Try out following example by putting the source code in test.php script.

```
<?php
  if( $_POST["name"] || $_POST["age"] ) {
    if (preg_match("/[^A-Za-z'-]/",$_POST['name'] )) {
      die ("invalid name and name should be alpha");
    }
    echo "Welcome ". $_POST['name']. "<br />";
    echo "You are ". $_POST['age']. " years old.";

    exit();
  }
?>
<html>
  <body>
```

```
    <form action = "<?php $_PHP_SELF ?>" method = "POST">
      Name: <input type = "text" name = "name" />
      Age: <input type = "text" name = "age" />
      <input type = "submit" />
    </form>

  </body>
</html>
```

It will produce the following result −



The $_REQUEST variable

The PHP $_REQUEST variable contains the contents of both $_GET, $_POST, and $_COOKIE. We will discuss $_COOKIE variable when we will explain about cookies.

The PHP $_REQUEST variable can be used to get the result from form data sent with both the GET and POST methods.

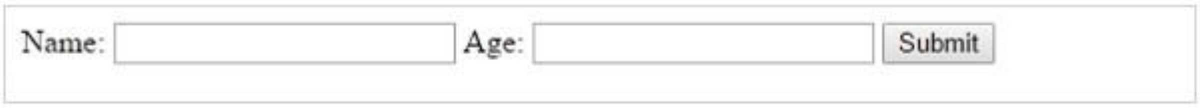
Try out following example by putting the source code in test.php script.

```
<?php
  if( $_REQUEST["name"] || $_REQUEST["age"] ) {
    echo "Welcome ". $_REQUEST['name']. "<br />";
    echo "You are ". $_REQUEST['age']. " years old.";
    exit();
  }
?>
<html>
  <body>

    <form action = "<?php $_PHP_SELF ?>" method = "POST">
      Name: <input type = "text" name = "name" />
      Age: <input type = "text" name = "age" />
      <input type = "submit" />
    </form>

  </body>
```

```
</html>
```

Here $_PHP_SELF variable contains the name of self script in which it is being called.

It will produce the following result −



PHP - Cookies

Cookies are text files stored on the client computer and they are kept of use tracking purpose. PHP transparently supports HTTP cookies.

There are three steps involved in identifying returning users −

- Server script sends a set of cookies to the browser. For example name, age, or identification number etc.
- Browser stores this information on local machine for future use.
- When next time browser sends any request to web server then it sends those cookies information to the server and server uses that information to identify the user.

This chapter will teach you how to set cookies, how to access them and how to delete them.

The Anatomy of a Cookie

Cookies are usually set in an HTTP header (although JavaScript can also set a cookie directly on a browser). A PHP script that sets a cookie might send headers that look something like this −

```
HTTP/1.1 200 OK
Date: Fri, 04 Feb 2000 21:03:38 GMT
Server: Apache/1.3.9 (UNIX) PHP/4.0b3
Set-Cookie: name=xyz; expires=Friday, 04-Feb-07 22:03:38 GMT;
            path=/; domain=tutorialspoint.com
Connection: close
Content-Type: text/html
```

As you can see, the Set-Cookie header contains a name value pair, a GMT date, a path and a domain. The name and value will be URL encoded. The expires field is an instruction to the browser to "forget" the cookie after the given time and date.

If the browser is configured to store cookies, it will then keep this information until the expiry date. If the user points the browser at any page that matches the path and domain of the cookie, it will resend the cookie to the server.The browser's headers might look something like this −

```
GET / HTTP/1.0
Connection: Keep-Alive
User-Agent: Mozilla/4.6 (X11; I; Linux 2.2.6-15apmac ppc)
Host: zink.demon.co.uk:1126
Accept: image/gif, */*
Accept-Encoding: gzip
Accept-Language: en
Accept-Charset: iso-8859-1,*,utf-8
Cookie: name=xyz
```

A PHP script will then have access to the cookie in the environmental variables $_COOKIE or $HTTP_COOKIE_VARS[] which holds all cookie names and values. Above cookie can be accessed using $HTTP_COOKIE_VARS["name"].

Setting Cookies with PHP

PHP provided **setcookie()** function to set a cookie. This function requires upto six arguments and should be called before <html> tag. For each cookie this function has to be called separately.

```
setcookie(name, value, expire, path, domain, security);
```

Here is the detail of all the arguments −

- **Name** − This sets the name of the cookie and is stored in an environment variable called HTTP_COOKIE_VARS. This variable is used while accessing cookies.
- **Value** − This sets the value of the named variable and is the content that you actually want to store.
- **Expiry** − This specify a future time in seconds since 00:00:00 GMT on 1st Jan 1970. After this time cookie will become

inaccessible. If this parameter is not set then cookie will automatically expire when the Web Browser is closed.

- **Path −** This specifies the directories for which the cookie is valid. A single forward slash character permits the cookie to be valid for all directories.
- **Domain −** This can be used to specify the domain name in very large domains and must contain at least two periods to be valid. All cookies are only valid for the host and domain which created them.
- **Security −** This can be set to 1 to specify that the cookie should only be sent by secure transmission using HTTPS otherwise set to 0 which mean cookie can be sent by regular HTTP.

Following example will create two cookies **name** and **age** these cookies will be expired after one hour.

```
<?php
   setcookie("name", "John Watkin", time()+3600, "/","", 0);
   setcookie("age", "36", time()+3600, "/", "",  0);
?>
<html>

   <head>
      <title>Setting Cookies with PHP</title>
   </head>

   <body>
      <?php echo "Set Cookies"?>
   </body>

</html>
```

Accessing Cookies with PHP

PHP provides many ways to access cookies. Simplest way is to use either $_COOKIE or $HTTP_COOKIE_VARS variables. Following example will access all the cookies set in above example.

```
<html>
```

```
  <head>
    <title>Accessing Cookies with PHP</title>
  </head>

  <body>

    <?php
      echo $_COOKIE["name"]. "<br />";

      /* is equivalent to */
      echo $HTTP_COOKIE_VARS["name"]. "<br />";

      echo $_COOKIE["age"] . "<br />";

      /* is equivalent to */
      echo $HTTP_COOKIE_VARS["age"] . "<br />";
    ?>

  </body>
</html>
```

You can use **isset()** function to check if a cookie is set or not.

```
<html>

  <head>
    <title>Accessing Cookies with PHP</title>
  </head>

  <body>

    <?php
      if( isset($_COOKIE["name"]))
        echo "Welcome " . $_COOKIE["name"] . "<br />";

      else
        echo "Sorry... Not recognized" . "<br />";
    ?>

  </body>
</html>
```

Deleting Cookie with PHP

Officially, to delete a cookie you should call setcookie() with the name argument only but this does not always work well, however, and should not be relied on.

It is safest to set the cookie with a date that has already expired −

```
<?php
  setcookie( "name", "", time()- 60, "/","", 0);
  setcookie( "age", "", time()- 60, "/","", 0);
?>
<html>

  <head>
    <title>Deleting Cookies with PHP</title>
  </head>

  <body>
    <?php echo "Deleted Cookies" ?>
  </body>

</html>
```

PHP - Sessions

An alternative way to make data accessible across the various pages of an entire website is to use a PHP Session.

A session creates a file in a temporary directory on the server where registered session variables and their values are stored. This data will be available to all pages on the site during that visit.

The location of the temporary file is determined by a setting in the **php.ini** file called **session.save_path**. Before using any session variable make sure you have setup this path.

When a session is started following things happen −

- PHP first creates a unique identifier for that particular session which is a random string of 32 hexadecimal numbers such as 3c7foj34c3jj973hjkop2fc937e3443.

- A cookie called **PHPSESSID** is automatically sent to the user's computer to store unique session identification string.
- A file is automatically created on the server in the designated temporary directory and bears the name of the unique identifier prefixed by sess_ ie sess_3c7foj34c3jj973hjkop2fc937e3443.

When a PHP script wants to retrieve the value from a session variable, PHP automatically gets the unique session identifier string from the PHPSESSID cookie and then looks in its temporary directory for the file bearing that name and a validation can be done by comparing both values.

A session ends when the user loses the browser or after leaving the site, the server will terminate the session after a predetermined period of time, commonly 30 minutes duration.

Starting a PHP Session

A PHP session is easily started by making a call to the **session_start()** function.This function first checks if a session is already started and if none is started then it starts one. It is recommended to put the call to **session_start()** at the beginning of the page.

Session variables are stored in associative array called **$_SESSION[]**. These variables can be accessed during lifetime of a session.

The following example starts a session then register a variable called **counter** that is incremented each time the page is visited during the session.

Make use of **isset()** function to check if session variable is already set or not.

Put this code in a test.php file and load this file many times to see the result −

```
<?php
  session_start();

  if( isset( $_SESSION['counter'] ) ) {
     $_SESSION['counter'] += 1;
  }else {
     $_SESSION['counter'] = 1;
  }

  $msg = "You have visited this page ".  $_SESSION['counter'];
  $msg .= "in this session.";
?>

<html>

  <head>
    <title>Setting up a PHP session</title>
  </head>

  <body>
    <?php  echo ( $msg ); ?>
  </body>

</html>
```

It will produce the following result −

```
You have visited this page 1in this session.
```

Destroying a PHP Session

A PHP session can be destroyed by **session_destroy()** function. This function does not need any argument and a single call can destroy all the session variables. If you want to destroy a single session variable then you can use **unset()** function to unset a session variable.

Here is the example to unset a single variable −

```
<?php
  unset($_SESSION['counter']);
?>
```

Here is the call which will destroy all the session variables −

```
<?php
  session_destroy();
?>
```

Turning on Auto Session

You don't need to call start_session() function to start a session when a user visits your site if you can set **session.auto_start** variable to 1 in **php.ini** file.

Sessions without cookies

There may be a case when a user does not allow to store cookies on their machine. So there is another method to send session ID to the browser.

Alternatively, you can use the constant SID which is defined if the session started. If the client did not send an appropriate session cookie, it has the form session_name=session_id. Otherwise, it expands to an empty string. Thus, you can embed it unconditionally into URLs.

The following example demonstrates how to register a variable, and how to link correctly to another page using SID.

```
<?php
  session_start();

  if (isset($_SESSION['counter'])) {
    $_SESSION['counter'] = 1;
  }else {
    $_SESSION['counter']++;
  }

  $msg = "You have visited this page ".  $_SESSION['counter'];
  $msg .= "in this session.";

  echo ( $msg );
?>

<p>
  To continue  click following link <br />

  <a  href = "nextpage.php?<?php echo htmlspecialchars(SID); ?>">
</p>
```

It will produce the following result −

```
You have visited this page 1in this session.
To continue click following link
```

The **htmlspecialchars()** may be used when printing the SID in order to prevent XSS related attacks.

global variable in PHP?

Global variables refer to any variable that is defined outside of the function. Global variables can be accessed from any part of the script i.e. inside and outside of the function. So, a global variable can be declared just like other variable but it must be declared outside of function definition.

Syntax:

```
$variable_name = data;
```

Below programs illustrate how to declare global variable.

Example 1:

```
<?php

// Demonstrate how to declare global variable


// Declaring global variable

$x = "you";

$y = "are";

$z = "good";


// Display value
// Concatenating String
echo $x.$y.$z;


?>
```

Output:

```
youaregood
```

Accessing global variable inside function: The ways to access the global variable inside functions are:

Using global keyword

Using array GLOBALS[var_name]: It stores all global variables in an array called $GLOBALS[var_name]. Var_name is the name of the variable. This array is also accessible from within functions and can be used to perform operations on global variables directly.

PHP | urlencode() Function

The urlencode() function is an inbuilt function in PHP which is used to encode the url. This function returns a string which consist all non-alphanumeric characters except -_. and replace by the percent (%) sign followed by two hex digits and spaces encoded as plus (+) signs.

Syntax:

```
string urlencode( $input )
```

Parameters: This function accepts single parameter $input which is used to hold the url to be encoded.

Return Value: This function returns an encoded string on success.

Below programs illustrate the urlencode() function in PHP:

Program 1:

```
<?php
// PHP program to illustrate urlencode function
echo urlencode("https://geeksforgeeks.org/") . "\n";
?>
```

Output:

```
https%3A%2F%2Fgeeksforgeeks.org%2F
```

PHP HTML Encode

HTML relies heavily on the use of tags and special characters. Some special characters in HTML contain special meaning that requires them to be used with caution.

For example, tags such as < and > are among the most widely used characters in HTML. Although they do not pose any threats on their own, when misused, they can break the entire web page.

Such HTML characters also stance a significant security flaw, especially in dynamic web applications. This can lead to the injection of malicious code such as JavaScript and form data.

MY LATEST VIDEOS

0 seconds of 12 secondsVolume 0%

The essence of this guide is to show you how you can use PHP to encode or “sanitize” HTML characters. Encoding such characters in dynamic websites will prevent Cross-Site Scripting and protect the web page from breaking.

What is Encoding?

Encoding refers to the process of converting reserved characters into HTML character entities. HTML character entities are expressed as &value; where the “value” represents an abbreviation or number for each character.

HTML offers a comprehensive collection of entities. However, we need only concern ourselves with four of them for encoding purposes:

1. < – &lt;
2. > – &gt;
3. & – &amp;
4. = – &quot;

Let us learn how we can use PHP to encode such characters.

PHP Encoding Functions

PHP has two main functions that you can use to encode HTML characters.

1. Htmlspecialchars()
2. Htmlentities()

The htmlspecialchars() functions encode the four main characters (above) while the htmentities() function will encode all the characters as possible.

PHP htmlspecialchars()

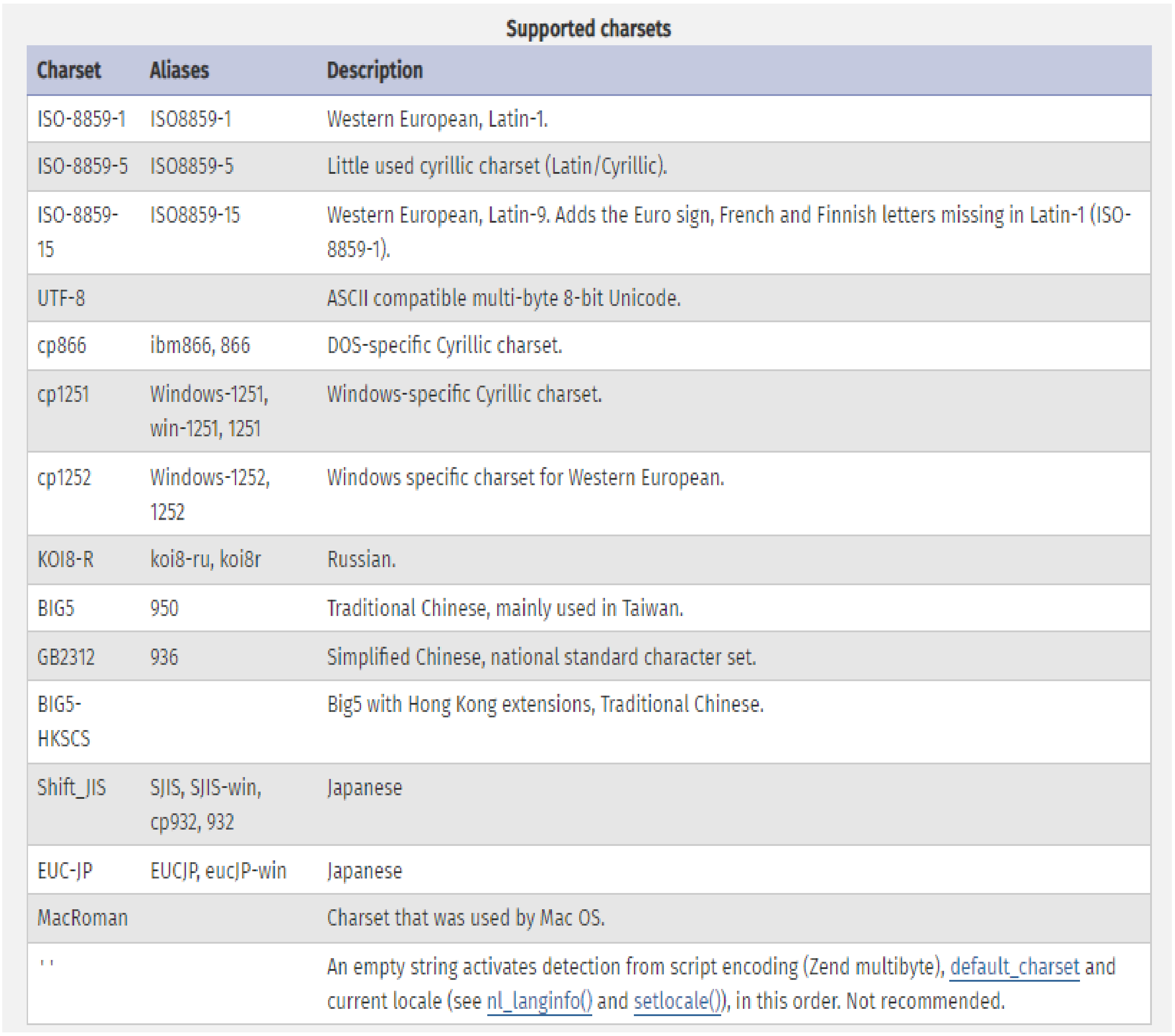
This function converts all special or reserved HTML characters to HTML entities. Although you can specify, the function will ignore single quotes by default.

The general syntax of the function is as shown:

```
htmlentities(string $string, int $flags bool);
```

The function accepts the string containing the HTML to be encoded. You can also specify flag values that allow you to tweak how the method operates.

PHP also allows you to specify the encoding method you wish to use for the HTML entities. The following image shows the supported charsets.



Supported charsets

Charset	Aliases	Description
ISO-8859-1	ISO8859-1	Western European, Latin-1.
ISO-8859-5	ISO8859-5	Little used cyrillic charset (Latin/Cyrillic).
ISO-8859-15	ISO8859-15	Western European, Latin-9. Adds the Euro sign, French and Finnish letters missing in Latin-1 (ISO-8859-1).
UTF-8		ASCII compatible multi-byte 8-bit Unicode.
cp866	ibm866, 866	DOS-specific Cyrillic charset.
cp1251	Windows-1251, win-1251, 1251	Windows-specific Cyrillic charset.
cp1252	Windows-1252, 1252	Windows specific charset for Western European.
KOI8-R	koi8-ru, koi8r	Russian.
BIG5	950	Traditional Chinese, mainly used in Taiwan.
GB2312	936	Simplified Chinese, national standard character set.
BIG5-HKSCS		Big5 with Hong Kong extensions, Traditional Chinese.
Shift_JIS	SJIS, SJIS-win, cp932, 932	Japanese
EUC-JP	EUCJP, eucJP-win	Japanese
MacRoman		Charset that was used by Mac OS.
''		An empty string activates detection from script encoding (Zend multibyte), default_charset and current locale (see nl_langinfo() and setlocale()), in this order. Not recommended.

The following example shows how to use the htmlspecialchars() method.

```
<?php
$str = "HTML uses < and > for <em>tags</em>";
echo htmlspecialchars($str);
?>
```

The above example will encode the HTML characters specified in the variable $str.

The output is as shown:

```
HTML uses &lt; and &gt; for &lt;em&gt;tags&lt;/em&gt;
```

If you want the function to process single and double quotes, you can use a flag as shown in the example below:

```
<?php
$str = "A single quote as 'and' will be ignored by default ";
echo htmlspecialchars($str, ENT_QUOTES);
?>
```

Once you run the above code, the function will process the single quotes and give an output as shown:

A single quote as 'and' will be ignored by default

PHP htmlentities()

We will look at the next encoding character is the PHP htmlentities(). This function converts all applicable HTML characters to HTML entities. It is a perfect choice when you need to process your HTML safely.

The general syntax of the function is as shown:

```
htmlentities(string $string, int flags);
```

The function is very similar to htmlspecialchars() except it processes all characters it can by default.

The following example shows you how to use the htmlentities() function.

```
<?php
$str = "<p>This is <i>valid</i> HTML code</p>";
echo htmlentities($str);
?>
```

The above code should return all the tags converted to entities as:

```
&lt;p&gt;This is &lt;i&gt;valid&lt;/i&gt; HTML code&lt;/p&gt;
```

Similar to the htmlspecialchars() function, it supports flags and encoding charset.

PHP - Files & I/O

This chapter will explain following functions related to files −

- Opening a file
- Reading a file
- Writing a file
- Closing a file

Opening and Closing Files

The PHP **fopen()** function is used to open a file. It requires two arguments stating first the file name and then mode in which to operate.

Files modes can be specified as one of the six options in this table.

Sr.No	Mode & Purpose
1	**r** Opens the file for reading only. Places the file pointer at the beginning of the file.
2	**r+** Opens the file for reading and writing. Places the file pointer at the beginning of the file.
3	**w**

	Opens the file for writing only. Places the file pointer at the beginning of the file. and truncates the file to zero length. If files does not exist then it attempts to create a file.
4	**w+** Opens the file for reading and writing only. Places the file pointer at the beginning of the file. and truncates the file to zero length. If files does not exist then it attempts to create a file.
5	**a** Opens the file for writing only. Places the file pointer at the end of the file. If files does not exist then it attempts to create a file.
6	**a+** Opens the file for reading and writing only. Places the file pointer at the end of the file. If files does not exist then it attempts to create a file.

If an attempt to open a file fails then **fopen** returns a value of **false** otherwise it returns a **file pointer** which is used for further reading or writing to that file.

After making a changes to the opened file it is important to close it with the **fclose()** function. The **fclose()** function requires a file pointer as its argument and then returns **true** when the closure succeeds or **false** if it fails.

Reading a file

Once a file is opened using **fopen()** function it can be read with a function called **fread()**. This function requires two arguments. These must be the file pointer and the length of the file expressed in bytes.

The files length can be found using the **filesize()** function which takes the file name as its argument and returns the size of the file expressed in bytes.

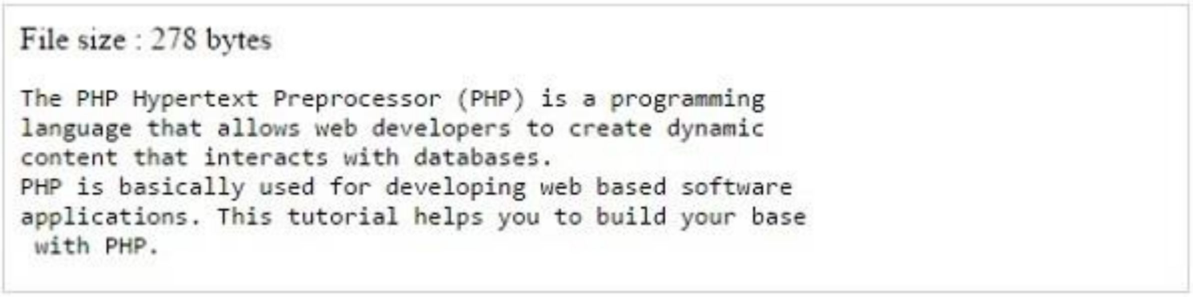
So here are the steps required to read a file with PHP.

- Open a file using **fopen()** function.
- Get the file's length using **filesize()** function.
- Read the file's content using **fread()** function.
- Close the file with **fclose()** function.

The following example assigns the content of a text file to a variable then displays those contents on the web page.

```
<html>

   <head>
      <title>Reading a file using PHP</title>
   </head>

   <body>

      <?php
         $filename = "tmp.txt";
         $file = fopen( $filename, "r" );

         if( $file == false ) {
            echo ( "Error in opening file" );
            exit();
         }

         $filesize = filesize( $filename );
         $filetext = fread( $file, $filesize );
         fclose( $file );

         echo ( "File size : $filesize bytes" );
         echo ( "<pre>$filetext</pre>" );
      ?>

   </body>
</html>
```

It will produce the following result −


File size : 278 bytes

```
The PHP Hypertext Preprocessor (PHP) is a programming
language that allows web developers to create dynamic
content that interacts with databases.
PHP is basically used for developing web based software
applications. This tutorial helps you to build your base
 with PHP.
```

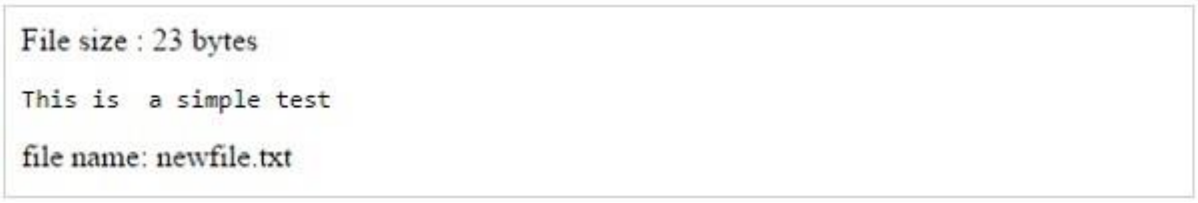
Writing a file

A new file can be written or text can be appended to an existing file using the PHP **fwrite()** function. This function requires two arguments specifying a **file pointer** and the string of data that is to be written. Optionally a third integer argument can be included to specify the length of the data to write. If the third argument is included, writing would will stop after the specified length has been reached.

The following example creates a new text file then writes a short text heading inside it. After closing this file its existence is confirmed using **file_exist()** function which takes file name as an argument

```
<?php
   $filename = "/home/user/guest/newfile.txt";
   $file = fopen( $filename, "w" );

   if( $file == false ) {
      echo ( "Error in opening new file" );
      exit();
   }
   fwrite( $file, "This is  a simple test\n" );
   fclose( $file );
?>
<html>

   <head>
      <title>Writing a file using PHP</title>
   </head>

   <body>

      <?php
         $filename = "newfile.txt";
         $file = fopen( $filename, "r" );
         if( $file == false ) {
            echo ( "Error in opening file" );
            exit();
         }
         
         $filesize = filesize( $filename );
         $filetext = fread( $file, $filesize );
         
         fclose( $file );
         
         echo ( "File size : $filesize bytes" );
         echo ( "$filetext" );
         echo("file name: $filename");
      ?>
      
   </body>
</html>
```

It will produce the following result −



```
File size : 23 bytes

This is  a simple test

file name: newfile.txt
```

PHP | Unique Features

- As PHP can do anything related to server-side programming which contains the backend of any web page,
  it holds a lot of unique features within it. The main purpose of PHP design is web development.

- Let us look into some of the unique features of PHP:

- Simple, Familiar and ease of use: Its popularly known for its simplicity, familiarity and easy to learn the language as the syntax is similar to that of ‘C’ or Pascal language.
  So the language is very logical and well organized general-purpose programming language. Even people with a normal programming background can easily understand and capture the use of language. PHP is very advantageous for new users as its a very reliable, fluent, organized, clean, demandable and efficient.

- The main strength of PHP is the availability of rich pre-defined functions. The core distribution helps the developers implement dynamic websites very easily with secured data. PHP applications are very easy to optimize.

- Loosely typed language: PHP encourages the use of variables without declaring its data types. So this is taken care at the execution time depending on the value assigned to the variable. Even the variable name can be changed dynamically.

- Flexibility: PHP is known for its flexibility and embedded nature as it can be well integrated with HTML, XML, Javascript and many more. PHP can run on multiple operating systems
  like Windows, Unix, Mac OS, Linux, etc. The PHP scripts can easily run on any device like laptops, mobiles, tablets, and computer. It is very comfortably integrated with various Databases. Desktop applications are created using advanced PHP features. The executable PHP can also be run on command-line as well as directly on the machine. Heavyweight applications can be created without a server or browser.
  It also acts as an excellent interface with relational databases.

- Open Source: All PHP frameworks are open sources, No payment is required for the users and its completely free. User can just download PHP and start using for their applications or projects. Even in companies, the total cost is reduced for software

development providing more
reliability and flexibility.

- It supports a popular range of databases like MySQL, SQLite, Oracle, Sybase, Informix, and PostgreSQL.
  PHP provides libraries to access these databases to interact with web servers. Developers are free to post errors, inspect codes and can contribute to code as well as bug fixing. Many frameworks like Codeignitor, Zend Framework, CakePHP make use of PHP.

- Even many popular content management systems like WordPress, Joomla and Drupal use PHP as prime language.
  Because of the above reasons many web hosting companies and Internet Service providers prefers PHP.

- Cross-platform compatibility: PHP is multi-platform and known for its portability as it can run on any operating System and windows environments. The most common are XAMPP (Windows, Apache Server, MySQL, Perl, and PHP) and LAMP
  (Linux, Apache, MySQL, PHP). As PHP is platform-independent, it’s very easy to integrate with various databases and other technologies without re-implementation. It effectively saves a lot of energy, time and money.

- Error reporting and exceptions: PHP supports much errors reporting constants to generate errors and relevant warnings at run time. For
  example E_ERROR, E_WARNING, E_PARSE, E_STRICT.
  PHP5 supports exception handling which is used to throw errors which can be caught at any time.
- Active community support: PHP is very rich with many diverse online community developers to help beginners for web-based applications. These worldwide volunteers contribute many features as well as new versions for PHP libraries. Even they contribute a translation in different languages to help out programmers. There is a bundle of third-party open-source libraries which provide basic functionalities. Even the documentation given by the official site

helps in implementing new features providing access to a variety of creative imagination.

- Fast and efficient performance: Users generally prefer fast loading websites.
  For any web development, speed becomes an important aspect which is taken care of by PHP.

- PHP scripts are faster than other scripting languages like ASP.NET, PERL, and JSP. The memory manager of PHP 7 is very optimized and fast as compared to older versions of PHP. Even connecting to the database and loading of required data from tables, are faster than other programming languages. It provides a built-in module for easy and efficient database management system. The high speed of PHP is advantageous for users for its server administration and mail functionality. Also, it supports session management and removing of unwanted memory allocation.

- Maintenance: When dealing with big projects, maintenance of code is also an important aspect of the web development process. There are many PHP frameworks for example MVC (Model View Controller) which makes development and maintenance of code easier. Files belonging to the different module are maintained separately.

- Third-party application support and security: Many PHP's predefined functions support data encryption options keeping it more secure. Even the users can use third-party applications to secure data.
- Real time access monitoring: PHP also provides a summary of user's recent logging accesses.

- Memory and CPU usage information: PHP can provide memory usage information from functions like memory_get_usage() or memory_get_peak_usage(), which can help the developers optimize their code. In the similar way, the

CPU power consumed by any script can be retrieved for further optimization.

- Object oriented features: PHP supports object-oriented programming features, resulting in increased speed and introducing added features like data encapsulation and inheritance at many levels.

- Magic Constants: PHP provides many built-in magic methods starting with __(double underscore) which are called during specific events.

- For example directory path
- (__DIR__), class name (__CLASS__), namespace (__NAMESPACE__), function name (__FUNCTION__), method name (__METHOD__), line number (__LINE__), file path (__FILE__).

- Regular expression: PHP provides regular expression functions with parsing and pattern matching methods.

- PDO Class: PHP Data Objects are created by PDO class which gives a good abstraction layer for database drivers. The PDO Classes are enriched with functions which are database independent. It means the same functions are used for similar actions for different databases without re-development as long as it supports PDO. In this way, the application becomes more portable saving lot of time and effort. Use of PDO helps the application from SQL injection attacks.