PHP Global Variables - Superglobals

Several predefined variables in PHP are "superglobals", which means that they are always accessible, regardless of scope - and you can access them from any function, class or file without having to do anything special. Superglobals were introduced in PHP 4.1.0, and are built-in variables that are always available in all scopes.

The PHP superglobal variables are:

- $GLOBALS
- $_SERVER
- $_REQUEST
- $_POST
- $_GET
- $_FILES
- $_ENV
- $_COOKIE
- $_SESSION

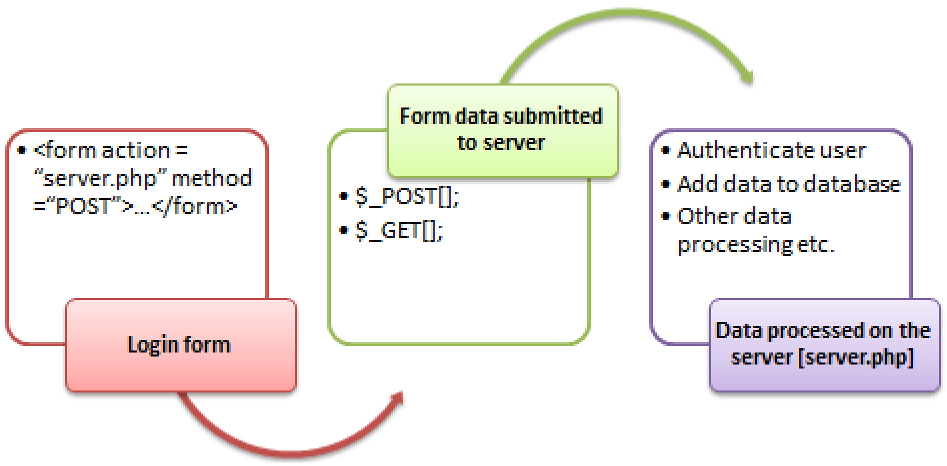
PHP Registration Form using GET, POST Methods with Example

What is Form?

When you login into a website or into your mail box, you are interacting with a form.

Forms are used to get input from the user and submit it to the web server for processing.

The diagram below illustrates the form handling process.



<form action = "server.php" method ="POST">...</form>
Login form
Form data submitted to server
$_POST[];
$_GET[];
Authenticate user
Add data to database
Other data processing etc.
Data processed on the server [server.php]

A form is an HTML tag that contains graphical user interface items such as input box, check boxes radio buttons etc.

The form is defined using the <form>...</form> tags and GUI items are defined using form elements such as input.

When and why we are using forms?

- Forms come in handy when developing flexible and dynamic applications that accept user input.
- Forms can be used to edit already existing data from the database

Create a form

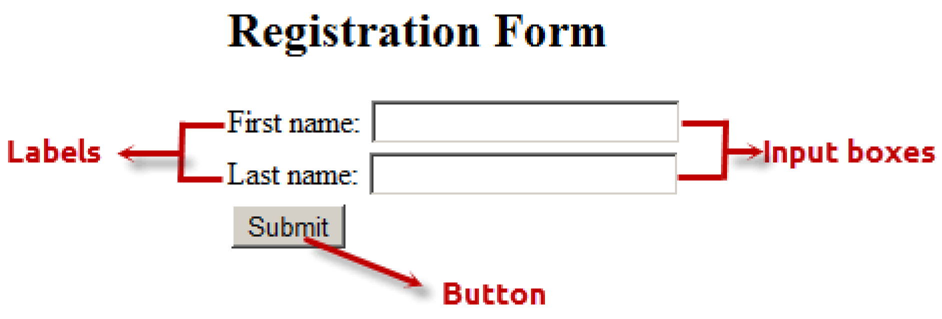
We will use HTML tags to create a form. Below is the minimal list of things you need to create a form.

- Opening and closing form tags <form>…</form>
- Form submission type POST or GET
- Submission URL that will process the submitted data
- Input fields such as input boxes, text areas, buttons,checkboxes etc.

The code below creates a simple registration form

```
<html>
<head>
        <title>Registration Form</title>
        </head>
<body>

  <h2>Registration Form</h2>
  <form action="registration_form.php" method="POST"> First name:

     <input type="text" name="firstname"> <br> Last name:

     <input type="text" name="lastname">

     <input type="submit" value="Submit">

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

Viewing the above code in a web browser displays the following form.



HERE,

- <form…>…</form> are the opening and closing form tags
- action="registration_form.php" method="POST"> specifies the destination URL and the submission type.
- First/Last name: are labels for the input boxes
- <input type="text"…> are input box tags
- <br> is the new line tag
- <input type="submit" value="Submit"> is the button that when clicked submits the form to the server for processing

Submitting the form data to the server

The action attribute of the form specifies the submission URL that processes the data. The method attribute specifies the submission type.

PHP POST method

- This is the built in PHP super global array variable that is used to get values submitted via HTTP POST method.
- The array variable can be accessed from any script in the program; it has a global scope.

- This method is ideal when you do not want to display the form post values in the URL.
- A good example of using post method is when submitting login details to the server.

It has the following syntax.

```
<?php
 $_POST['variable_name'];
?>
```

HERE,

- “$_POST[…]” is the PHP array
- “'variable_name'” is the URL variable name.

PHP GET method

- This is the built in PHP super global array variable that is used to get values submitted via HTTP GET method.
- The array variable can be accessed from any script in the program; it has a global scope.
- This method displays the form values in the URL.
- It’s ideal for search engine forms as it allows the users to book mark the results.

It has the following syntax.

```
<?php
$_GET['variable_name'];
?>
```

HERE,

- “$_GET[…]” is the PHP array
- “'variable_name'” is the URL variable name.

GET vs POST Methods

POST	GET
Values not visible in the URL	Values visible in the URL
Has not limitation of the length of the values since they are submitted via the body of HTTP	Has limitation on the length of the values usually 255 characters. This is because the values are displayed in the URL. Note the upper limit of the characters is dependent on the browser.
Has lower performance compared to Php_GET method due to time spent encapsulation the Php_POST values in the HTTP body	Has high performance compared to POST method dues to the simple nature of appending the values in the URL.
Supports many different data types such as string, numeric, binary etc.	Supports only string data types because the values are displayed in the URL
Results cannot be book marked	Results can be book marked due to the visibility of the values in the URL
More secure	Less secure

$_GET and *$_POST* are Superglobal variables in PHP which used to collect data from HTML form and URL.

The example below contains an HTML form with two input fields, and a submit button:

Example:

```
<html>

<body>



<form action="registration.php" method="post">

Name: <input type="text" name="name">

Email: <input type="text" name="email">
```

```
<input type="submit">

</form>



</body>

</html>
```

When the user fills out and submits the form, then form data will be sent to PHP file: called *registration.php*.

registration.php page has following code to print submitted data:

```
<html>

<body>



Welcome <?php echo $_POST["name"]; ?>!

Your email address is <?php echo $_POST["email"]; ?>



</body>

</html>
```

Program Output:

```
Welcome Alex!

Your email address is alex@example.com.
```

Form GET/POST method and PHP $_GET/$_POST

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

- The GET Method
- The POST Method

PHP $_GET Variable

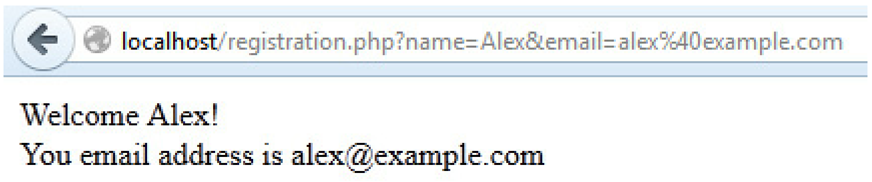
In PHP, the *$_GET* variable is used to collect values from HTML forms using method *get*.

Information sent from an HTML form with the GET method is displayed in the browser's address bar, and it has a limit on the amount of information to send.

Example:

```
<html>

<body>



<form action="registration.php" method="get">

Name: <input type="text" name="name">

Email: <input type="text" name="email">

<input type="submit">

</form>



</body>

</html>
```

When the user clicks on the "Submit button", the URL will be something like this:



```
localhost/registration.php?name=Alex&email=alex%40example.com
```

```
Welcome Alex!
You email address is alex@example.com
```

registration.php looks like this:

```
<html>

<body>
```

```
Welcome <?php echo $_GET["name"]; ?>!

Your email address is <?php echo $_GET["email"]; ?>



</body>

</html>
```

When to use method="get"?

- The variable names and values will be visible in URL if HTML forms submitted by the GET method.
- The GET method is restricted to send up to *1024 characters* only.
- When you submit sensitive information like passwords then should not use this method.
- GET method can't be used, to send binary data like images and Word documents.
- GET method data can be accessed using PHP QUERY_STRING environment variable.
- PHP $_GET associative array is used to access all the sent information by GET method.

PHP $_POST Variable

In PHP, the *$_POST* variable is used to collect values from HTML forms using method *post*.

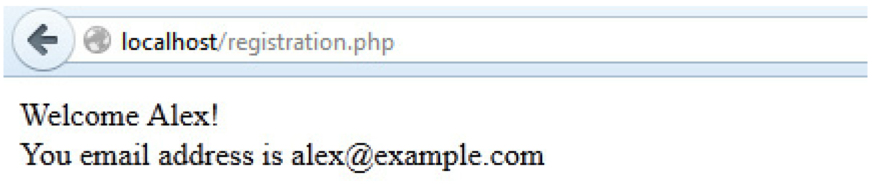
Information sent from a form with the POST method is invisible and has no limits on the amount of information to send.

Example:

```
<html>

<body>



<form action="registration.php" method="post">

Name: <input type="text" name="name">
```

```
Email: <input type="text" name="email">

<input type="submit">

</form>



</body>

</html>
```

When the user clicks on the "Submit button", the URL will be something like this:



localhost/registration.php

Welcome Alex!
You email address is alex@example.com

registration.php looks like this:

Example:

```
<html>

<body>



Welcome <?php echo $_POST["name"]; ?>!

Your email address is <?php echo $_POST["email"]; ?>



</body>

</html>
```

When to use method="post"?

- 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.

- PHP $_POST associative array is used to access all the sent information by POST method.
- Variables are not visible in the URL so users can't bookmark your page.

PHP - A Simple HTML Form

The example below displays a simple HTML form with two input fields and a submit button:

Example

```
<html>
<body>

<form action="welcome.php" method="post">
Name: <input type="text" name="name"><br>
E-mail: <input type="text" name="email"><br>
<input type="submit">
</form>

</body>
</html>
```

When the user fills out the form above and clicks the submit button, the form data is sent for processing to a PHP file named "welcome.php". The form data is sent with the HTTP POST method.

To display the submitted data you could simply echo all the variables. The "welcome.php" looks like this:

```
<html>
<body>

Welcome <?php echo $_POST["name"]; ?><br>
Your email address is: <?php echo $_POST["email"]; ?>

</body>
</html>
```

The output could be something like this:

```
Welcome John
Your email address is john.doe@example.com
```

The same result could also be achieved using the HTTP GET method:

Example

```
<html>
<body>

<form action="welcome_get.php" method="get">
Name: <input type="text" name="name"><br>
E-mail: <input type="text" name="email"><br>
<input type="submit">
</form>

</body>
</html>
```

and "welcome_get.php" looks like this:

```
<html>
<body>

Welcome <?php echo $_GET["name"]; ?><br>
Your email address is: <?php echo $_GET["email"]; ?>

</body>
</html>
```

GET vs. POST

Both GET and POST create an array (e.g. array(key1 => value1, key2 => value2, key3 => value3, ...)). This array holds key/value pairs, where keys are the names of the form controls and values are the input data from the user.

Both GET and POST are treated as $_GET and $_POST. These are superglobals, which means that they are always accessible, regardless of scope - and you can access them from any function, class or file without having to do anything special.

$_GET is an array of variables passed to the current script via the URL parameters.

$_POST is an array of variables passed to the current script via the HTTP POST method.

When to use GET?

Information sent from a form with the GET method is **visible to everyone** (all variable names and values are displayed in the URL). GET also has limits on the amount of information to send. The limitation is about 1024 characters. However, because the variables are displayed in the URL, it is possible to bookmark the page. This can be useful in some cases.

GET may be used for sending non-sensitive data.

Note: GET should NEVER be used for sending passwords or other sensitive information!

When to use POST?

Information sent from a form with the POST method is **invisible to others** (all names/values are embedded within the body of the HTTP request) and has **no limits** on the amount of information to send.

Moreover POST supports advanced functionality such as support for multi-part binary input while uploading files to server.

However, because the variables are not displayed in the URL, it is not possible to bookmark the page.

The PHP $_REQUEST Variable

PHP $_REQUEST is used to collect data after submitting an HTML form.

The example below shows a form with an input field and a submit button. When a user submits the data by clicking on "Submit", the form data is sent to the file specified in the action attribute of the <form> tag. In this example, we point to this file itself for processing form data. If you wish to use another PHP file to process form data, replace that with the filename of your choice. Then, we can use the super global variable $_REQUEST to collect the value of the input field:

The *$_REQUEST* variable contains the contents of $_GET, $_POST, and $_COOKIE.

Example:

```
<html>

<body>



Welcome <?php echo $_REQUEST["name"]; ?>!

Your email address is <?php echo $_REQUEST["Email"]; ?>
```

```
</body>

</html>
```

Summary

- Forms are used to get data from the users
- Forms are created using HTML tags
- Forms can be submitted to the server for processing using either POST or GET method
- Form values submitted via the POST method are encapsulated in the HTTP body.
- Form values submitted via the GET method are appended and displayed in the URL

PHP 5 Form Handling Example

In this tutorial, we will create a simple HTML form with different types of inputs and then write PHP code for processing the form-data when submitted.

We will also learn some useful tips that can be utilized while processing the form-data like checking for empty input values, validating fields etc. So let's get started.

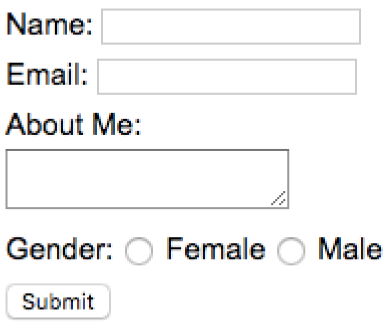
The HTML Form

We will be creating an HTML form with 2 text fields for name and email, 1 textarea for the user to provide a self-description(more like about me) and a radio button set for asking the user's gender.

Below we have the HTML code for the form.

```
<html>
  <body>

    <form action="form-handler.php" method="POST">

      Name: <input type="text" name="name"> <br/>
      Email: <input type="text" name="email"> <br/>

      About Me:<br/>
      <textarea name="aboutme"></textarea> <br/>

```

```
        Gender:
        <input type="radio" name="gender" value="female"> Female
        <input type="radio" name="gender" value="male"> Male
        <br/>
        <input type="submit" name="submit" value="Submit">
      </form>

   </body>
</html>
```



The PHP Code

In the above form we are asking the user for 4 different inputs, let's see how we can fetch the submitted values for the fields in PHP.

```
<?php

// getting the value of name field
$name = $_POST["name"];
// getting the value of the email field
$email = $_POST["email"];
// getting the value of the aboutme field
$aboutme = $_POST["aboutme"];
// getting the value of the radio buttons
$gender = $_POST["gender"];

?>
```

Easy, right? Yes, to access the form-data for different types of HTML form elements, all you need is `$_POST`.

But an HTML form is a great entry point for hackers to play around by entering script or some malicious code into the input fields to cause some error/issue in your PHP script.

To tackle with it, it's good to put some validations in the PHP code for validating the user inputs submitted in the form-data.

PHP Form Validation

Now we will learn some basic validations that can be easily applied to the submitted form-data to validate it before performing any action on it.

```
<?php

// getting the value of name field
$name = $_POST["name"];
// check if name is empty or not
if(empty($name)) {
   echo "Name is required";
}

?>
```

In the code above we are checking whether the user has entered **name** value in the form or not, similarly you can put a check on all the mandatory form fields.

To validate email address, there is a special function available in PHP which we can use to validate email addresses. The function is `filter_var($email, FILTER_VALIDATE_EMAIL)`, let's see how it works.

```
<?php

// getting the value of the email field
$email = $_POST["email"];

// checking if the email value is valid
if(!filter_var($email, FILTER_VALIDATE_EMAIL)) {
   echo "Email value invalid";
}

?>
```

The `filter_var()` function returns `true` for a valid email address and returns `false` for an invalid email address.

Validating a form in PHP depends on your requirements too. For example, if you

don't have any mandatory fields in your form, then you don't have to worry about checking whether the submitted values are empty or not.

If you have email address field, we suggest you validate it.

You can even add more validations like checking the input for malicious codes like `<script>` tags using regular expressions.

What is `$_SERVER["PHP_SELF"]`?

Sometimes we can avoid having an extra PHP file to handle/process the form-data and can include the PHP code in the file with the HTML form itself.

In such scenarios, we will have to submit the form to the same webpage again, and we can use `$_SERVER["PHP_SELF"]` as the form **action**.

```
<form action="<?php echo $_SERVER["PHP_SELF"];  ?>" method="POST">
```

What this superglobal does is that it returns the filename of the current webpage, which then acts as the action script.

But as this returns the existing filename from the URL, you must be a little careful because users may inject some unwanted code from the URL, so to avoid it, we can use the `htmlspecialchars()` function to convert any special character in the string(URL in this case) into HTML entities.

So you should use,

```
<form action="<?php echo htmlspecialchars($_SERVER["PHP_SELF"]);  ?>" method="POST">
```

PHP isset() function

PHP isset() function is used to check if a variable has been set or not.
This can be useful to check the submit button is clicked or not.
The isset() function will return true or false value. The isset() function returns true if variable is set and not null.

PHP isset() function syntax

isset(“variable”);

PHP isset() Example

```
<?php


$str = 'meera';
var_dump(isset($str));

```

```
?>
```

The output is :

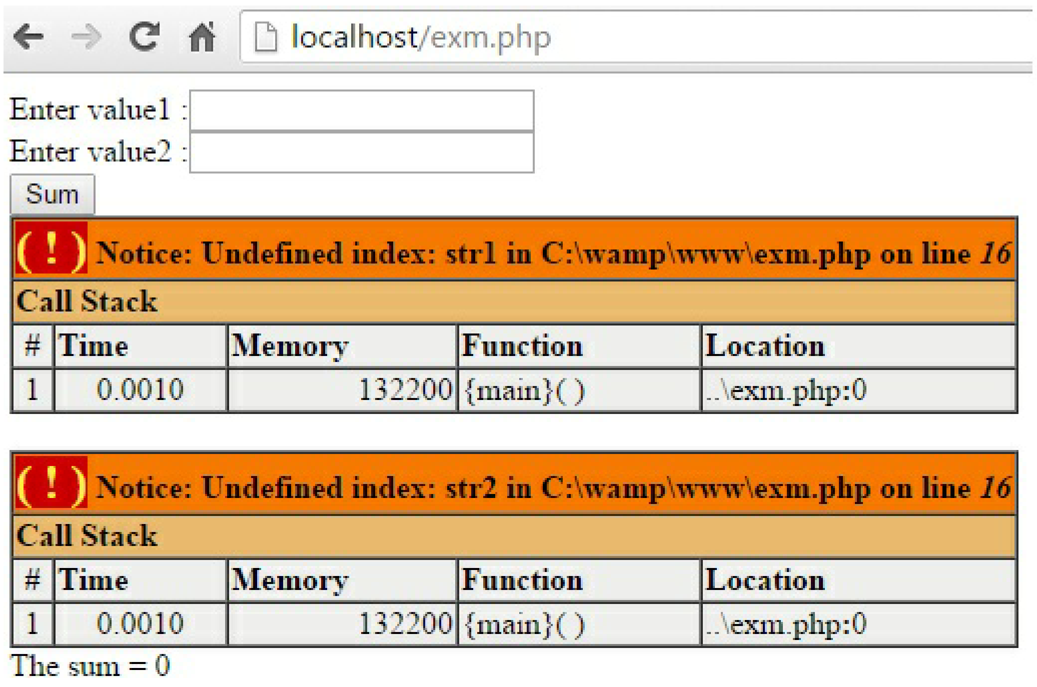
boolean true

lets another php form example to understand more about isset() function.

Here, we are do sum of two values, design php form with two textbox and one button control like :

```
<html>
<head>
<title>PHP isset() example</title>
</head>
<body>

<form method="post">

Enter value1 :<input type="text" name="str1"><br/>
Enter value2 :<input type="text" name="str2"><br/>
<input type="submit" value="Sum" name="Submit1">

<?php
$sum=$_POST["str1"] + $_POST["str2"];
echo "The sum = ". $sum;
?>

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

When we run above php example while first time page load it will show error like below:



localhost/exm.php

Enter value1 :

Enter value2 :

Sum

(!) **Notice: Undefined index: str1 in C:\wamp\www\exm.php on line *16***

Call Stack

#	Time	Memory	Function	Location
1	0.0010	132200	{main}( )	..\exm.php:0

(!) **Notice: Undefined index: str2 in C:\wamp\www\exm.php on line *16***

Call Stack

#	Time	Memory	Function	Location
1	0.0010	132200	{main}( )	..\exm.php:0

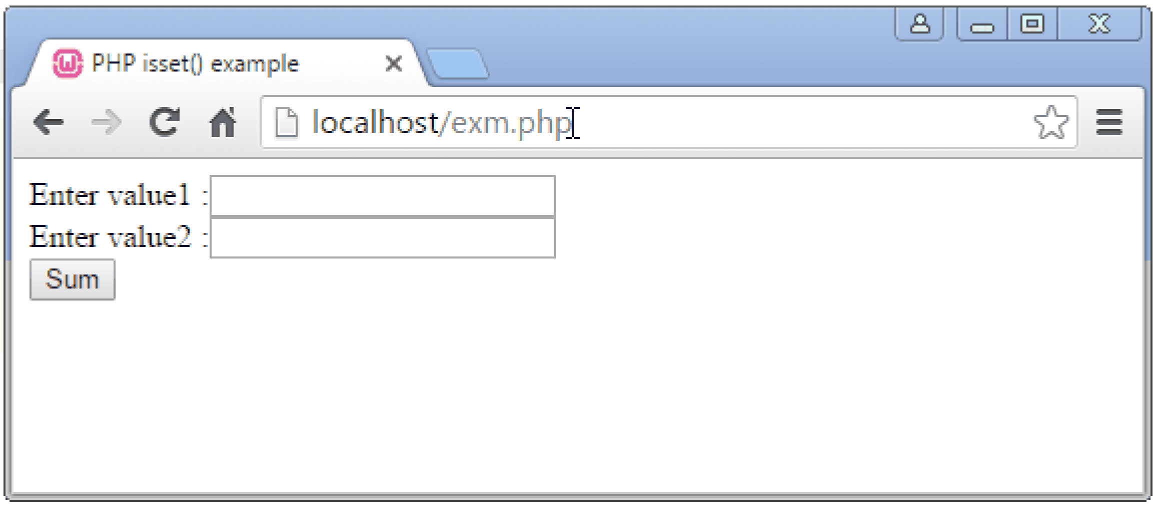
The sum = 0

PHP isset() function example

In above code add isset() function code to remove above error while page loading.

```
1  <html>
2  <head>
3  <title>PHP isset() example</title>
4  </head>
5  <body>
6
7  <form method="post">
8
9  Enter value1 :<input type="text" name="str1"><br/>
10 Enter value2 :<input type="text" name="str2"><br/>
11 <input type="submit" value="Sum" name="Submit1"><br/><br/>
12
13 <?php
14 if(isset($_POST["Submit1"]))
15 {
16 $sum=$_POST["str1"] + $_POST["str2"];
17 echo "The sum = ". $sum;
18
19 }
20 ?>
21
22 </form>
23 </body>
24 </html>
```

The output of isset() function php example is :



PHP isset() function example

PHP Global Variables - Superglobals

PHP $GLOBALS

$GLOBALS is a PHP super global variable which is used to access global variables from anywhere in the PHP script (also from within functions or methods).

PHP stores all global variables in an array called $GLOBALS[*index*]. The *index* holds the name of the variable.

The example below shows how to use the super global variable $GLOBALS:

Example

```
<?php
$x = 75;
$y = 25;

function addition() {
   $GLOBALS['z'] = $GLOBALS['x'] + $GLOBALS['y'];
}

addition();
echo $z;
?>
```

In the example above, since z is a variable present within the $GLOBALS array, it is also accessible from outside the function!

PHP $_SERVER

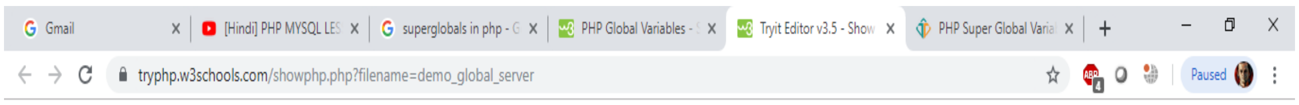
$_SERVER is a PHP super global variable which holds information about headers, paths, and script locations.

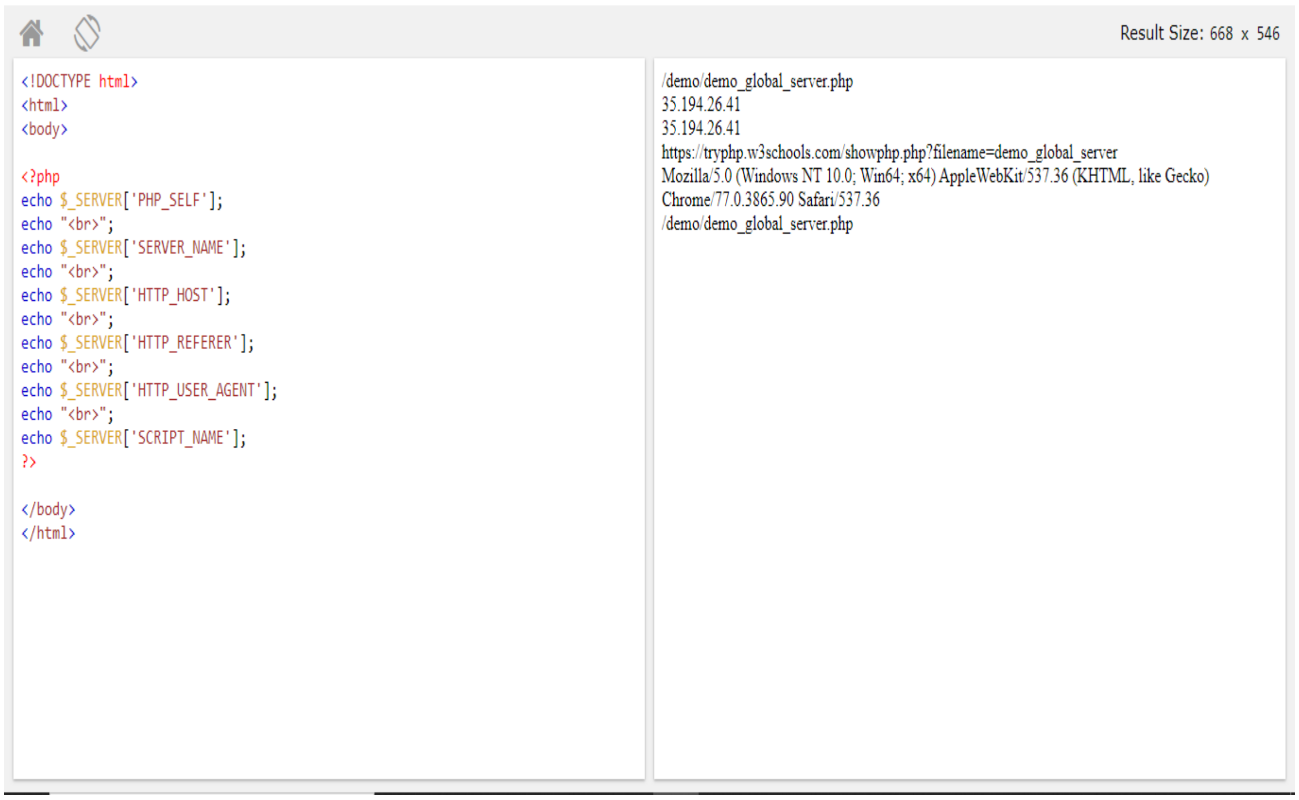
The example below shows how to use some of the elements in $_SERVER:

Example

```
<?php
echo $_SERVER['PHP_SELF'];
echo "<br>";
echo $_SERVER['SERVER_NAME'];
echo "<br>";
echo $_SERVER['HTTP_HOST'];
echo "<br>";
echo $_SERVER['HTTP_REFERER'];
echo "<br>";
echo $_SERVER['HTTP_USER_AGENT'];
```

```php
echo "<br>";
echo $_SERVER['SCRIPT_NAME'];


?>
```





```php
<!DOCTYPE html>
<html>
<body>

<?php
echo $_SERVER['PHP_SELF'];
echo "<br>";
echo $_SERVER['SERVER_NAME'];
echo "<br>";
echo $_SERVER['HTTP_HOST'];
echo "<br>";
echo $_SERVER['HTTP_REFERER'];
echo "<br>";
echo $_SERVER['HTTP_USER_AGENT'];
echo "<br>";
echo $_SERVER['SCRIPT_NAME'];
?>

</body>
</html>
```

```
/demo/demo_global_server.php
35.194.26.41
35.194.26.41
https://tryphp.w3schools.com/showphp.php?filename=demo_global_server
Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/77.0.3865.90 Safari/537.36
/demo/demo_global_server.php
```

The following table lists the most important elements that can go inside $_SERVER:

Element/Code	**Description**
$_SERVER['PHP_SELF']	Returns the filename of the currently executing script

$_SERVER['GATEWAY_INTERFACE']	Returns the version of the Common Gateway Interface (CGI) the server is using
$_SERVER['SERVER_ADDR']	Returns the IP address of the host server
$_SERVER['SERVER_NAME']	Returns the name of the host server (such as www.w3schools.com)
$_SERVER['SERVER_SOFTWARE']	Returns the server identification string (such as Apache/2.2.24)
$_SERVER['SERVER_PROTOCOL']	Returns the name and revision of the information protocol (such as HTTP/1.1)
$_SERVER['REQUEST_METHOD']	Returns the request method used to access the page (such as POST)
$_SERVER['REQUEST_TIME']	Returns the timestamp of the start of the request (such as 1377687496)
$_SERVER['QUERY_STRING']	Returns the query string if the page is accessed via a query string
$_SERVER['HTTP_ACCEPT']	Returns the Accept header from the current request
$_SERVER['HTTP_ACCEPT_CHARSET']	Returns the Accept_Charset header from the current request (such as utf-8,ISO-8859-1)

$_SERVER['HTTP_HOST']	Returns the Host header from the current request
$_SERVER['HTTP_REFERER']	Returns the complete URL of the current page (not reliable because not all user-agents support it)
$_SERVER['HTTPS']	Is the script queried through a secure HTTP protocol
$_SERVER['REMOTE_ADDR']	Returns the IP address from where the user is viewing the current page
$_SERVER['REMOTE_HOST']	Returns the Host name from where the user is viewing the current page
$_SERVER['REMOTE_PORT']	Returns the port being used on the user's machine to communicate with the web server
$_SERVER['SCRIPT_FILENAME']	Returns the absolute pathname of the currently executing script
$_SERVER['SERVER_ADMIN']	Returns the value given to the SERVER_ADMIN directive in the web server configuration file (if your script runs on a virtual host, it will be the value defined for that virtual host) (such as someone@w3schools.com)
$_SERVER['SERVER_PORT']	Returns the port on the server machine being used by the web server for communication

	(such as 80)
$_SERVER['SERVER_SIGNATURE']	Returns the server version and virtual host name which are added to server-generated pages
$_SERVER['PATH_TRANSLATED']	Returns the file system based path to the current script
$_SERVER['SCRIPT_NAME']	Returns the path of the current script
$_SERVER['SCRIPT_URI']	Returns the URI of the current page

PHP String Function Examples

1) PHP strtolower() function

The strtolower() function returns string in lowercase letter.

Syntax

1. string strtolower (string $string)

Example

1. <?php
2. $str="My name is KHAN";

```
$str=strtolower($str);
echo $str;
?>
```

Output:

```
my name is khan
```

2) PHP strtoupper() function

The strtoupper() function returns string in uppercase letter.

Syntax

```
string strtoupper ( string $string )
```

Example

```
<?php
$str="My name is KHAN";
$str=strtoupper($str);
echo $str;
?>
```

Output:

```
MY NAME IS KHAN
```

3) PHP ucfirst() function

The ucfirst() function returns string converting first character into uppercase. It doesn't change the case of other characters.

Syntax

```
string ucfirst ( string $str )
```

Example

```
<?php
$str="my name is KHAN";
$str=ucfirst($str);
echo $str;
?>
```

Output:

```
My name is KHAN
```

4) PHP lcfirst() function

The lcfirst() function returns string converting first character into lowercase. It doesn't change the case of other characters.

Syntax

```
string lcfirst ( string $str )
```

Example

```
<?php
$str="MY name IS KHAN";
$str=lcfirst($str);
echo $str;
?>
```

Output:

```
mY name IS KHAN
```

5) PHP ucwords() function

The ucwords() function returns string converting first character of each word into uppercase.

Syntax

```
string ucwords ( string $str )
```

Example

```
<?php
$str="my name is Sonoo jaiswal";
$str=ucwords($str);
echo $str;
?>
```

Output:

```
My Name Is Sonoo Jaiswal
```

6) PHP strrev() function

The strrev() function returns reversed string.

Syntax

```
string strrev ( string $string )
```

Example

```
<?php
$str="my name is Sonoo jaiswal";
$str=strrev($str);
echo $str;
?>
```

Output:

```
lawsiaj oonoS si eman ym
```

7) PHP strlen() function

The strlen() function returns length of the string.

Syntax

```
int strlen ( string $string )
```

Example

```
<?php
$str="my name is Sonoo jaiswal";
$str=strlen($str);
echo $str;
?>
```

Output:

```
24
```

PHP String Functions

strlen($str)

This function returns the length of the string or the number of characters in the string including whitespaces.

```
<?php


$str = "Welcome to Studytonight";


// using strlen in echo method
echo "Length of the string is: ". strlen($str);
```

```
?>
```

```
Length of the string is: 23
```

---

str_word_count($str)

This function returns the number of words in the string. This function comes in handy in form field validation for some simple validations.

```php
<?php

$str = "Welcome to Studytonight";

// using str_word_count in echo method
echo "Number of words in the string are: ". str_word_count($str);

?>
```

```
Number of words in the string are: 3
```

---

strrev($str)

This function is used to reverse a string.

Let's take an example and see,

```php
<?php

$str = "Welcome to Studytonight";

// using strrev in echo method
echo "Reverse: ". strrev($str);

?>
```

```
Reverse: thginotydutS ot emocleW
```

---

strpos($str, $text)

This function is used to find the position of any text/word in a given string. Just like an array, string also assign index value to the characters stored in it, starting from zero.

```
<?php

$str = "Welcome to Studytonight";

// using strpos in echo method
echo "Position of 'Studytonight' in string: ". strpos($str, 'Studytonight');

?>
```

```
Position of 'Studytonight' in string: 11
```

---

str_replace($replacethis, $replacewith, $str)

This function is used to replace a part of the string with some text. While using this function, the first argument is the part of string that you want to replace, second argument is the new text you want to include, and the last argument is the string variable itself.

Let's take an example,

```
<?php

$str = str_replace("Studytonight", "Studytonight.com", "Welcome to Studytonight");

echo $str;

?>
```

```
Welcome to Studytonight.com
```

---

ucwords($str)

This function is used for formatting the string. This function converts first letter/character of every word in the string to uppercase.

Let's take an example and see,

```
<?php

$str = "welcome to studytonight";
```

```
echo ucwords($str);

?>
```

```
Welcome To Studytonight
```

---

strtoupper($str)

To convert every letter/character of every word of the string to uppercase, one can use `strtoupper()` method.

```
<?php

$str = "welcome to studytonight";

echo strtoupper($str);

?>
```

```
WELCOME TO STUDYTONIGHT
```

---

strtolower($str)

This function is used to convert every letter/character of a string to lowercase.

```
<?php

$str = "WELCOME TO STUDYTONIGHT";

echo strtolower($str);

?>
```

```
welcome to studytonight
```

---

str_repeat($str, $counter)

This function is used to repeat a string a given number of times. The first argument is the string and the second argument is the number of times the string should be repeated.

```
<?php

$str = "Studytonight";

echo str_repeat($str, 4);

?>
```

```
StudytonightStudytonightStudytonightStudytonight
```

---

substr($str, $start, $length)

This function is used to take out a part of the string(substring), starting from a particular position, of a particular length.

The first argument is the string itself, second argument is the starting index of the substring to be exracted and the third argument is the length of the substring to be extracted.

```
<?php

$str = "Welcome to Studytonight";

echo substr($str, 11, 12);

?>
```

```
Studytonight
```

---

trim($str, charlist)

This function is used to remove extra whitespaces from beginning and the end of a string. The second argument `charlist` is optional. We can provide a list of character, just like a string, as the second argument, to trim/remove those characters from the main string.

```
<?php

$str1 = "   Hello World   ";
```

```
echo trim($str1) . "<br/>";

$str2 = "Hello Hello";

echo trim($str2,"Heo");

?>
```

```
Hello World
llo Hell
```

As you can see in the output, additional spaces from the beginning and end are removed and in the second case, the characters specified are removed from the beginning and the end of the string.

---

explode(separator, $str, $limit)

This function is used to break a string, create an array of the broken parts of the string and return the array. The first argument, **separator** defines where to break the string from. It can be a space, hiphen(-) or any other character.

The second argument of this function is the string itself and the third argument is the **limit**, which specifies the number of array elements to return. the third argument is optional.

Let's have an example,

```
<?php

$str = "Its a beautiful day";

// we can directly print the result of explode
print_r(explode(" ", $str));

?>
```

```
Array (
[0] => Its
[1] => a
[2] => beautiful
[3] => day
)
```

In the example above, we have provided **space** as separator to break the string and return an array.

If we provide the third argument **limit** as well, we can limit the number of array elements returned. For example, if we provide 2 as the third argument, then we will only get 2 elements in the array, the first two.

---

implode(separator, $arr)

This function is used to form a string using the array elements from the array provided and join them using the **separator**.

Let's take an example,

```
<?php

$arr = array("Its", "a", "beautiful", "day");

// <br> is used to jump to next line
echo implode(" ", $arr) . "<br>";
echo implode("-", $arr) . "<br>";
echo implode("/", $arr) . "<br>";

?>
```

```
Its a beautiful day
Its-a-beautiful-day
Its/a/beautiful/day
```

PHP chr() Function

The PHP chr() function is used to generate a single byte string from a number. In another words we can say that it returns a character from specified ASCII value.

Syntax:

1. string chr (int $bytevalue);

Parameter	Description	Required/Optional
Asci	An ASCII value	Required

Example 1

```
<?php
$char =52;
echo "Your character is :".$char;
echo "<br>"."By using 'chr()' function your value is: ".chr($char);//  decimal Value
?>
```

Output:

```
Your character is :52
By using 'chr()' function your value is: 4
```

Example 2

```
<?php
$char =052;
echo "Your character is :".$char;
echo "<br>"."By using 'chr()' function your value is: ".chr($char); // Octal Value
?>
```

Output:

```
Your character is :42
By using 'chr()' function your value is: *
```

PHP string ord() Function

PHP string ord() is predefined function. It is used to convert first byte of string to a value between 0 and 255. It returns ASCII value.

Syntax:

```
int ord ( string $string );
```

Parameter	Description	Required/Optional
String	Specify string value	required

Example 1

```
<?php
```

Parameter	Description	Required/Optional
String	Specify the string to check.	Required
Charlist	Specify character to remove from the string. o "\0" : NULL o "\t" : tab o "\n" : new line o "\x0B" : vertical tab o "\r" : carriage return o " " : ordinary white space	Optional

```php
echo "Your Value is: A"."<br>";
echo "By using 'ord()' Function:".ord("A")."<br>";
?>
```

Output:

```
Your Value is: A
By using 'ord()' Function:80
```

Example 2

```php
<?php
echo "Your Value is: PHP"."<br>";
echo "By using 'ord()' Function:".ord("PHP")."<br>";
?>
```

Output:

```
Your Value is: PHP
By using 'ord()' Function:80
```

PHP string ltrim() function

PHP string ltrim() function is predefined function. It is often used to remove whitespace from both sides of a string or other character from the left side of a string.

Syntax:

```php
ltrim(string,charlist);
```

Example 1

```
<?php
$str = "    Hello PHP!";
echo "Without ltrim() Function: " . $str;
echo "<br>";
echo "With ltrim() function : " . ltrim($str);
?>
```

Output:

```
Without ltrim() Function: Hello PHP!
With ltrim() function : Hello PHP!
```

Example 2

```
<?php
$str = "Hello      PHP      Javatpoint!";
echo "Without ltrim() Function: " . $str;
echo "<br>";
echo "With ltrim() function: " . ltrim($str);
?>
```

Output:

```
Without ltrim() Function: Hello PHP Javatpoint!
With ltrim() function: Hello PHP Javatpoint!
```

PHP string rtrim() Function

PHP rtrim() is predefined function which is used to remove whitespace or character form the right side of a string.

Syntax:

```
rtrim(string,charlist);
```

Parameter	Description	Requi
String	Specify the string to check.	requ

Charlist	Specify character to remove Following character o "\0" - NULL o "\t" - tab o "\n" - new line o "\x0B" - vertical tab o "\r" - carriage return o " " - ordinary white space	Opti

Example 1

```
<?php
$str = "Hello PHP!    ";
echo "Without rtrim() function: " . $str;
echo "<br>";
echo "With rtrim() function:" . rtrim($str);
?>
```

Output:

```
Without rtrim() function: Hello PHP!
With rtrim() function:Hello PHP!
```

PHP strcmp() Function

Example

Compare two strings (case-sensitive):

```
<?php
echo strcmp("Hello world!","Hello world!");
?>
```

Definition and Usage

The strcmp() function compares two strings.

Note: The strcmp() function is binary-safe and case-sensitive.

Tip: This function is similar to the strncmp() function, with the difference that you can specify the number of characters from each string to be used in the comparison with strncmp().

Syntax

strcmp(*string1*,*string2*)

Parameter Values

Parameter	Description
string1	Required. Specifies the first string to compare
string2	Required. Specifies the second string to compare

Technical Details

Return Value:	This function returns: • 0 - if the two strings are equal • <0 - if string1 is less than string2 • >0 - if string1 is greater than string2
PHP Version:	4+

Example

Compare two strings (case-sensitive = Hello and hELLo will not output the same):

```
<!DOCTYPE html>

<html>

<body>
```

```
<?php

echo strcmp("Hello world!", “Hello world!");

?>

<p>If this function returns 0, the two strings are equal.</p>

</body>

</html>
```

Output:
0
If this function returns 0, the two strings are equal.

Example

Different return values:

```
<?php
echo strcmp("Hello world!","Hello world!"); // the two strings are equal
echo strcmp("Hello world!","Hello"); // string1 is greater than string2
echo strcmp("Hello world!","Hello world! Hello!"); // string1 is less than string2
?>
```

Output:

```
0
7
-7
```

PHP string strcasecmp() function

PHP string strcasecmp() is predefine function. It is used to compare two given string. It is binary safe and case-insensitive. It is similar to the strncasecmp() function.

It returns:

- If the two strings are equal: [0]
- If string1 is less than string2: [< 0]
- If string1 is greater than string2: > 0

Syntax:

1. strcasecmp(string1,string2);

Parameter	Description	Required

string 1	Specify first string to compare.	Required
String2	Specify the second string to compare.	Reuired.

Example 1

```
<?php
$str1 = "JavaTPOINT";
$str2 = "JAVAtpoint";
echo "Your first string is:".$str1;
echo "<br>";
echo "Your second string is:".$str2;
echo "<br>";
echo strcasecmp("$str1","$str2");
?>
```

Output:

```
Your first string is:JavaTPOINT
Your second string is:JAVAtpoint
0
```

PHP strpos() Function

The strops() is in-built function of PHP. It is used to find the position of the first occurrence of a string inside another string or substring in a string.

Syntax:

```
int strpos ( string $haystack , mixed $needle [, int $offset = 0 ] );
```

Parameter	Description	Required/
String	Specify the string to search.	Start
Find	Specify the string to find.	Required
Start	Specify where to begin the search.	Optional

This function will help us to find the numeric position of the first occurrence of needle in the haystack string

Note: The strops() function is case-sensitive and binary-safe.

Example 1

```
<?php
 $str1="Hello Php";
  $str2="Hello Php javatpoint!";
  echo "First string is: ".$str1;
  echo "<br>";
  echo "First string is: ".$str2;
  echo "<br>";
  echo "By using 'strpos()' function:".strpos("$str1,$str2","php");
  //$str1 and $str2 'Php'first letter is upper case so output is nothing.
  // It is case-sensitive
?>
```

Output:

```
First string is: Hello Php
First string is: Hello Php javatpoint!
By using 'strpos()' function:
```

PHP String strstr() function

The strstr() function is an in-built function of PHP. It is a **case-sensitive** function which finds the first occurrence of a string. The strstr() is mainly used to search the first occurrence of a string inside another string and displays some part of the latter starting from the first occurrence of the former in latter.

It is a **binary-safe** function, means a function that can perform on binary files without modifying the content of the file. The strstr() function is similar to stristr() function, but the only difference is that - stristr() is case-insensitive function whereas strstr() is case-sensitive.

*Note: **The strstr() is** binary-safe **and** case-sensitive **function.***

Syntax

The syntax for the PHP strstr() function is given below, which consists three parameters.

```
stristr ($string, $search, $before_search)
```

Parameters

$string (required): $string is a mandatory parameter which specifies the string to be searched. In other words, it is the main string parameter in which $search value is searched.

$search (required): The next **mandatory** parameter of this function is $search. It specifies that string which is going to be searched in $string parameter. If this parameter contains a number or integer value rather than a string, then it will search for character matching ASCII value for that number.

$before_search (optional): It is the last and **optional** parameter of strstr() function which specifies the Boolean value, whose default value is **FALSE**. If we set it to TRUE, then it will return the part of the string before the first occurrence of the search parameter.

Return Values

The PHP strstr() returns the remaining string (from the matching point), or it will return **FALSE** if the string which we are searching for is not found.

Technical Details

PHP version support	PHP 4+ versions support this function.
Return Values	The strstr() returns rest of the string or returns False if the string for search is no
ChangeLog	o In strstr(), $before_search parameter was added in PHP 5.3

Examples of strstr()

There are some examples given below that will help us to learn the practical use of this function.

//Returns remaining string after search found
Input:
$string1 = "Hello! Good Morning everyone", $search1 = "Good";
Output: Morning everyone

//case-sensitive, returns nothing
Input:
$string = "Hello! Good Morning everyone ", $search = "HELLO ";
Output:

//Returns initial string when search found
Input:
$string = "Hello! Good Morning everyone", $search = "Good", before_search = true;
Output: Hello!

//Passing ASCII value of r which is 114
Input:

$string = "I want to travel the world ", $search = "114", before_search = true;

Output: I want to t

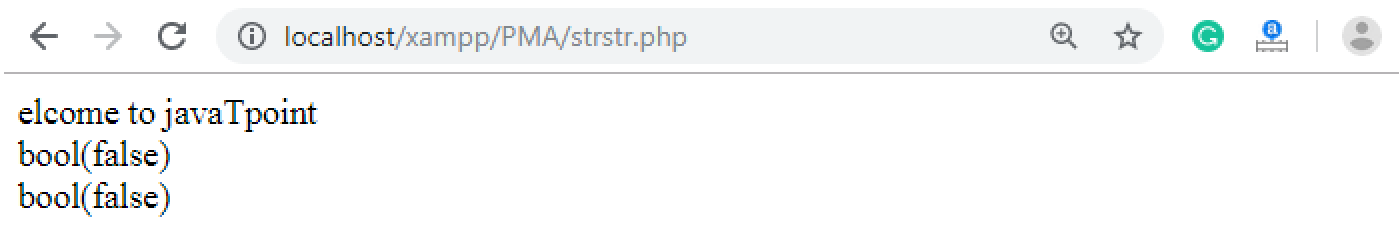
Below some detailed examples are given. With the help of these examples, we can understand the use of this function in a much better way.

Example 1

It is a simple example of strstr() which shows that it is a case-sensitive function, so it returns FALSE if the string does not match. It will return the rest of the string from where the $search variable will find in the string.

```
<?php
   $string = "Welcome to javaTpoint";
   $search1 = "e";
   echo strstr($string, $search1);

   echo '</br>';
   $search2 = "JAVA";        //case-sensitive
   var_dump(strstr($string, $search2));

   echo '</br>';
   var_dump(strstr($string, "WeLcOmE"));
?>
```

Output:



```
elcome to javaTpoint
bool(false)
bool(false)
```

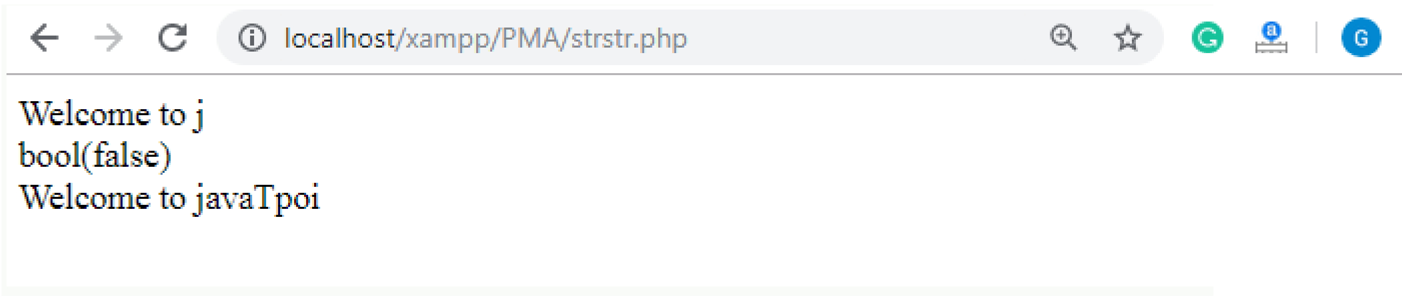
Example 2

In this below example, we will use a third parameter $before_search, whose default value is **FALSE**. Here, we will set it to **TRUE**, which will return the part of the string before the first occurrence of the $search parameter. If the search string is not found, then it will return the Boolean value **false**.

```
<?php
$string = "Welcome to javaTpoint website.";
   $before_search = true;
   $search1 = "a";
   echo strstr($string, $search1, $before_search);

   echo '</br>';
```

```
    $search2 = "E";
    var_dump(strstr($string, $search2, $before_search));

    echo '</br>';
    $search3 = "nt";
    echo strstr($string, $search3, $before_search);
?>
```

Output:



```
Welcome to j
bool(false)
Welcome to javaTpoi
```

stristr() function in PHP

The stristr() is an in-built function of PHP which is used to search the first occurrence of the string inside another string, and it returns rest of the string if the string is present. It is a **binary-safe** function (Binary safe function means - a function which can be performed on binary file without modifying the content of the file).

The stristr() is a **case-insensitive** function which is similar to the strstr(). Both functions are used to search a string inside another string. The only difference between them is that stristr() is **case-insensitive** whereas strstr() is **case-sensitive** function.

Note: ***This function is*** *binary-safe* ***and*** *case-insensitive* ***function. In stristr() function 'I' stands for insensitive.***

Syntax

The syntax for the PHP stristr() function is as follows:

```
stristr ($string, $search, $before_search)
```

Parameter

$string (required): This parameter is a **mandatory** parameter which specifies the string which is to be searched, means it is the main string in which $search (discussed next) value is searched.

$search (required): This parameter is also **mandatory** as $string. This parameter specifies the string which is going to be searched in $string. If this parameter is a number or integer value rather than a string, then it will use it as ASCII value.

$before_search (optional): This parameter is an **optional** parameter which specifies the Boolean value whose default value is **FALSE**. If we set it to TRUE, then it will return the part of the string before the first occurrence of the search parameter.

Return Values

The PHP stristr() returns the remaining string (starts from the matching point), or it will return FALSE, if the string which we are searching for is not found.

Technical Details

PHP version support	PHP 4 and above versions support this function.
Return Values	It returns rest of the string or returns False if the string to be searched is not fo
ChangeLog	o In PHP 5.3, the before_search parameter was added in this function. o The stristr() function became a binary-safe function in PHP 4.3.

Examples

Below are some examples through which you can learn the practical implementation of the stristr() function in the program.

Input:
$string = "Hello PHP! ", $search = "PHP ";
Output: PHP!

Input:
$string = "Hello PHP! ", $search = "p ", before_search = true;   //case-insensitive
Output: Hello

Input:
$string = "Hello PHP! ", $search = "K ", before_search = true;
Output:

Following are some detailed examples which are given below -

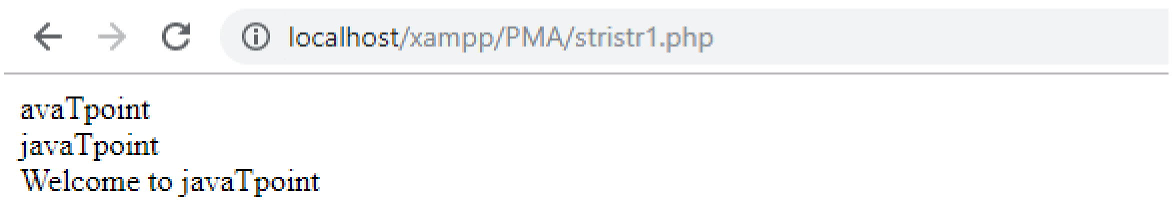
Example 1

It is the simple example of stristr() which shows that it is case-insensitive function and return the rest of the string where the $search variable will find.

```
<?php
$string = "Welcome to javaTpoint";
  $search1 = "a";
```

```
    echo stristr($string, $search1);

    echo '</br>';
    $search2 = "J";        //case-insensitive
    echo stristr($string, $search2);

    echo '</br>';
    echo stristr($string, "WELcoME");
?>
```

Output:



```
avaTpoint
javaTpoint
Welcome to javaTpoint
```

PHP Math

PHP provides many predefined math constants and functions that can be used to perform mathematical operations.

PHP Math: abs() function

The abs() function returns absolute value of given number. It returns an integer value but if you pass floating point value, it returns a float value.

Syntax

```
number abs ( mixed $number )
```

Example

```
<?php
echo (abs(-7)."<br/>"); // 7 (integer)
echo (abs(7)."<br/>"); //7 (integer)
echo (abs(-7.2)."<br/>"); //7.2 (float/double)
?>
```

Output:

```
7
7
7.24
```

PHP Math: ceil() function

The ceil() function rounds fractions up.

Syntax

```
float ceil ( float $value )
```

Example

```php
<?php
echo (ceil(3.3)."<br/>");// 4
echo (ceil(7.333)."<br/>");// 8
echo (ceil(-4.8)."<br/>");// -4
?>
```

Output:

```
4
8
-4
```

PHP Math: floor() function

The floor() function rounds fractions down.

Syntax

```
float floor ( float $value )
```

Example

```php
<?php
echo (floor(3.3)."<br/>");// 3
echo (floor(7.333)."<br/>");// 7
echo (floor(-4.8)."<br/>");// -5
?>
```

Output:

```
3
7
-5
```

PHP Math: sqrt() function

The sqrt() function returns square root of given argument.

Syntax

```
float sqrt ( float $arg )
```

Example

```php
<?php
echo (sqrt(16)."<br/>");// 4
echo (sqrt(25)."<br/>");// 5
echo (sqrt(7)."<br/>");// 2.6457513110646
?>
```

Output:

```
4
5
2.6457513110646
```

PHP fmod() Function

PHP fmod() function is mathematical function, which is used to return the floating point remainder of the division of the argument.

Syntax:

```php
float fmod ( float $x , float $y );
```

Parameter	Description	Required/Optional
X	Specifiy the dividend	Required
Y	Specify the divisor	Required

Example 1

```php
<?php
$x = 7;
$y = 2;
echo "Your Given Nos is : $x=7, $y=2"
$result ="By using 'fmod()' Function your value is:".fmod($x,$y);
echo $result;
// $result equals 1, because 2 * 3 + 1 = 7
?>
```

Output:

```
By using 'fmod()' Function your value is:1
```

Example 2

```php
<?php
$x = 5.7;
$y = 1.3;
echo "Your Given Nos is : x=5.7, y=1.3";
echo "<br>"."By using 'fmod()' function your value is:".fmod($x, $y);
// $r equals 0.5, because 4 * 1.3 + 0.5 = 5.7
?>
```

Output:

```
Your Given Nos is : x=5.7, y=1.3
By using 'fmod()' Function your value is:0.5
```

PHP Max() Function

The max() function is used to find the highest value.

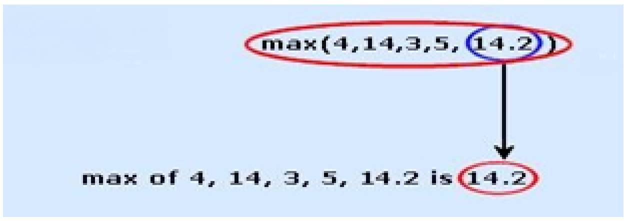
Syntax:

```php
max(array_values);
```

OR

```php
max(value1,value2,value3,value4...);
```

Name	Description
array_values	Specifies an array containing the values
value1,value2,value3,value4...	Specifies the values to compare (must be at least two values



Example1

```php
<?php
$num=max(4,14,3,5,14.2);
echo "Your number is =max(4,14,3,5,14.2)".'<br>';
echo "By using max function Your number is : ".$num;
?>
```

Output:

```
Your number is =max(4,14,3,5,14.2)
By using max function Your number is : 14.2
```

Example2

```php
<?php
$num=max(.1, .001, .2, -.5);
echo "Your number is =max(.1, .001, .2, -.5)".'<br>';
echo "By using max function Your number is : ".$num;
?>
```

Output:

```
Your number is =max(.1, .001, .2, -.5)
By using max function Your number is : 0.2
```

Min() function

PHP min() function is used to find the lowest value.

Syntax:

```
min(array_values);
           or
   min(value1,value2,value3...);
```

Parameter	Description
array_values	Specified array
value1,value2,value3...);	Specifies the two or more value (at least two numbers).

Example1

```php
<?php
$num=min(4,14,3,5,14.2);
echo "Your number is =min(4,14,3,5,14.2)".'<br>';
echo "By using min() function Your number is : ".$num;
?>
```

Output:

```
Your number is =min(4,14,3,5,14.2)
By using min function Your number is : 3
```

PHP pow() function

The pow() is a PHP mathematic function. It raises the first number to the power of the second number.

Syntax:

```
number pow ( number $base , number $exp )
```

Parameter	Description	Required /Optional
Base	Required. Specifies the base to use.	required
Exp	Required. Specifies the exponent.	required

Return Value: Returns a number (floating-point & Integer) which is equal to $base raised to the power of $exponent.

Example 1

```
<?php
$num=pow(3, 2);
echo "Your number is = pow (3, 2)".'<br>';
echo "By using pow function Your number is : ".$num;
?>
```

Output:

```
Your number is =pow(3, 2)
By using pow function Your number is : 9
```

Rand() function

The rand() function is used to generate random integer.

Syntax:

```
int rand ( void )
    or
```

```
int rand ( int $min , int $max )
```

Parameter	Description	Required/Optional
Min	Specifies the lowest number to returned.	Optional
Max	Specifies the highest number to be returned.	Optional

Example1

```
<?php
echo "Get Random number by using rand() function: ".(rand() . "<br>");
echo "Get Random number by using rand() function: ".(rand() . "<br>");
echo "<b>"."Note: Refresh page to get another randome value"."<b>";
?>
```

Output:

```
Get Random number by using rand() function: 81627923
Get Random number by using rand() function: 1857469033
Note: Refresh page to get another randome value
```

Example2

```
<?php
echo "To get random number b/w (rand(10,100)): ".(rand(10,100));
echo "<br>"."<br>"."Note: Refresh page to get new random number";
?>
```

Output:

```
To get random number b/w (rand(10,100)): 15
Note: Refresh page to get new random number
```

Round() function :

The round() function is used to find rounds a float number. It has three parameters and returns the round value.

Syntax:

```
float round($number, $precision, $mode);
```

Parameter	Description	Optional/Required
number	Specify the value to round	Required
Precision	Specify the number of decimal digits to round to.	Optional
Mode	Specify the mode in which rounding occurs. o PHP_ROUND_HALF_UP o PHP_ROUND_HALF_DOWN o PHP_ROUND_HALF_EVEN o PHP_ROUND_HALF_ODD	Optional

Example1

```php
<?php
echo "Befor using round() function : 3.96754,2";
echo "<br>"."After using round() function : ".(round(3.96754,2));
?>
```

Output:

```
Befor using round() function : 3.96754,2
After using round() function : 3.97
```

Example2

```php
<?php
echo "Befor using round() function : 7.045,2";
echo "<br>"."After using round() function : ".(round(7.045,2));
?>
```

Output:

```
Befor using round() function : 7.045,2
After using round() function : 7.05
```

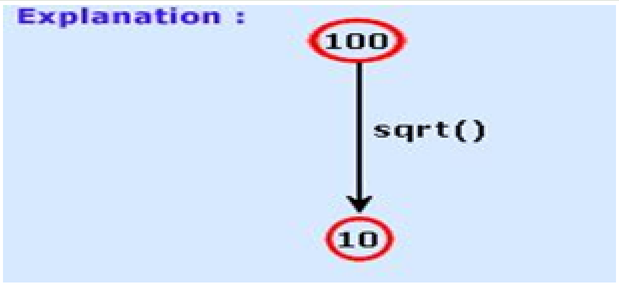
PHP sqrt() function

The sqrt() is mathematical function, used to calculate the square root of any number.

Syntax:

```
float sqrt(value)
```

Parameter	Description	Required /Optional
Number	Specifies a number	Required



Example1

```
<?php
$num=625;
echo "Your number is : $num".'<br>';
echo "By using sqrt function Your number is : ".sqrt ($num);
?>
```

Output:

```
Your number is : 625
By Using sqrt function Your number is : 25
```

PHP count() function

PHP count() function counts all elements in an array.

Syntax

```
int count ( mixed $array_or_countable [, int $mode = COUNT_NORMAL ] )
```

Example

```
<?php
$season=array("summer","winter","spring","autumn");
echo count($season);
?>
```

Output:

```
4
```

PHP - Sort Functions For Arrays

- sort() - sort arrays in ascending order
- rsort() - sort arrays in descending order
- asort() - sort associative arrays in ascending order, according to the value
- ksort() - sort associative arrays in ascending order, according to the key
- arsort() - sort associative arrays in descending order, according to the value
- krsort() - sort associative arrays in descending order, according to the key

---

Sort Array in Ascending Order - sort()

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

Example

```
<?php
$cars = array("Volvo", "BMW", "Toyota");
sort($cars);

$clength = count($cars);

for($x = 0; $x < $clength; $x++) {

    echo $cars[$x];

    echo "<br>";

}
?>
```

OUTPUT:

```
BMW
Toyota
Volvo
```

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

Example

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

```
?>
```

Sort Array in Descending Order - rsort()

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

Example

```
<?php
$cars = array("Volvo", "BMW", "Toyota");
rsort($cars);

$clength = count($cars);

for($x = 0; $x < $clength; $x++) {

   echo $cars[$x];

   echo "<br>";

}
?>
```

OUTPUT:

```
Volvo
Toyota
BMW
```

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

Example

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

Sort Array (Ascending Order), According to Value - asort()

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

Example

```
<?php
```

```
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");
asort($age);

foreach($age as $x => $x_value) {

  echo "Key=" . $x . ", Value=" . $x_value;

  echo "<br>";

}
?>
```

OUTPUT:

```
Key=Peter, Value=35
Key=Ben, Value=37
Key=Joe, Value=43
```

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

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

Example

```
<?php
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");
ksort($age);
?>
```

Sort Array (Descending Order), According to Value - arsort()

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

Example

```
<?php
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");
arsort($age);

foreach($age as $x => $x_value) {

  echo "Key=" . $x . ", Value=" . $x_value;
```

```
    echo "<br>";
}
?>
```

OUTPUT:

```
Key=Ben, Value=37
Key=Joe, Value=43
Key=Peter, Value=35
```

Sort Array (Descending Order), According to Key - krsort()

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

Example

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

OUTPUT:

```
Key=Peter, Value=35
Key=Joe, Value=43
Key=Ben, Value=37
```

PHP array_reverse() function

PHP array_reverse() function returns an array containing elements in reversed order.

Syntax

```
array array_reverse ( array $array [, bool $preserve_keys = false ] )
```

Example

```php
<?php
$season=array("summer","winter","spring","autumn");
$reverseseason=array_reverse($season);
foreach( $reverseseason as $s )
{
  echo "$s<br />";
}
?>
```

Output:

```
autumn
spring
winter
summer
```

PHP array_search() function

PHP array_search() function searches the specified value in an array. It returns key if search is successful.

Syntax

```php
mixed array_search ( mixed $needle , array $haystack [, bool $strict = false ] )
```

Example

```php
<?php
$season=array("summer","winter","spring","autumn");
$key=array_search("spring",$season);
echo $key;
?>
```

Output:

```
2
```

is_array($arr)

To check whether the provided data is in form of an array, we can use the is_array() function. It returns True if the variable is an array and returns False otherwise.

```php
<?php


$lamborghinis = array("Urus", "Huracan", "Aventador");


// using ternary operator
```

```
echo is_array($lamborghinis) ? 'Array' : 'not an Array';


$mycar = "Urus";


// using ternary operator
echo is_array($mycar) ? 'Array' : 'not an Array';


?>
```

```
Array
not an Array
```

---

in_array($var, $arr)

When using an array, we may often want to check whether a certain value is present in the array or not. For example, if get a list of certain cars, like we do in almost all our examples, to check if a certain car is added into the array, we can use the in_array function.

Let's take an example and see,

```
<?php


$lamborghinis = array("Urus", "Huracan", "Aventador");


// new concept car by lamborghini
$concept = "estoque";


echo in_array($concept, $lamborghinis) ? 'Added to the Lineup' : 'Not yet!'


?>
```

```
Not yet!
```

As we can see unlike the other functions above, this one takes **two arguments**, one is the value to be searched in the array, and the second one is the array itself.

print_r($arr)

Although this is not an array function, but it deserves a special mention here, as we can use this function to print the array in the most descriptive way possible. This function prints the complete representation of the array, along with all the keys and values.

```php
<?php

$lamborghinis = array("Urus", "Huracan", "Aventador");
print_r($lamborghinis);

?>
```

```
Array (
[0] => "Urus"
[1] => "Huracan"
[2] => "Aventador"
)
```

---

array_merge($arr1, $arr2)

If you want to combine two different arrays into a single array, you can do so using this function. It doesn't matter whether the arrays to be combined are of same type(indexed, associative etc) or different types, using this function we can combine them into one single array.

Let's take an example where we will merge an indexed array and an associative array.

```php
<?php

$hatchbacks = array(
        "Suzuki" => "Baleno",
        "Skoda" => "Fabia",
        "Hyundai" => "i20",
        "Tata" => "Tigor"
    );

// friends who own the above cars
$friends = array("Vinod", "Javed", "Navjot", "Samuel");

// let's merge the two arrays into one
$merged = array_merge($hatchbacks, $friends);

print_r($merged);

?>
```

```
Array (
[Suzuki] => Baleno
[Skoda] => Fabia
[Hyundai] => i20
[Tata] => Tigor
[0] => Vinod
[1] => Javed
[2] => Navjot
[3] => Samuel
)
```

array_reverse($arr)

This function is used to reverse the order of elements, making the first element last and last element first, and similarly rearranging other array elements.

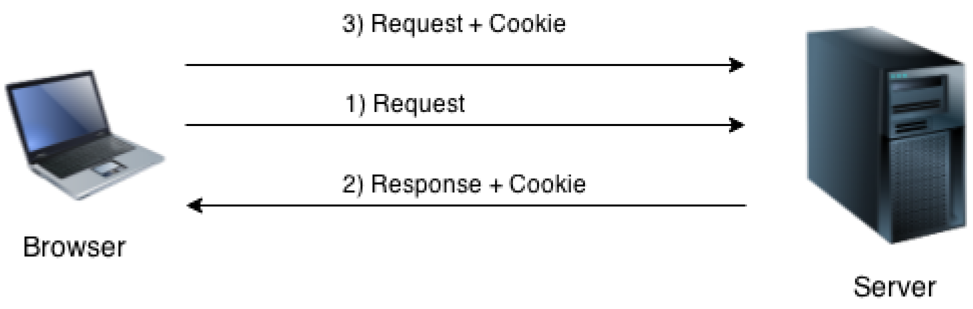
```php
<?php

$num = array(10, 20, 30, 40, 50);
// printing the array after reversing it
print_r(array_reverse($num));

?>
```

```
Array (
[0] => 50
[1] => 40
[2] => 30
[3] => 20
[4] => 10
)
```

PHP Cookie

PHP cookie is a small piece of information which is stored at client browser. It is used to recognize the user.

Cookie is created at server side and saved to client browser. Each time when client sends request to the server, cookie is embedded with request. Such way, cookie can be received at the server side.



In short, cookie can be created, sent and received at server end.

Note: PHP Cookie must be used before <html> tag.

PHP setcookie() function

PHP setcookie() function is used to set cookie with HTTP response. Once cookie is set, you can access it by $_COOKIE superglobal variable.

Syntax

```
1. bool setcookie ( string $name [, string $value [, int $expire = 0 [, string $path
2. [, string $domain [, bool $secure = false [, bool $httponly = false ]]]]]] )
```

Example

```
1. setcookie("CookieName", "CookieValue");/* defining name and value only*/
2. setcookie("CookieName", "CookieValue", time()+1*60*60);//using expiry in 1 hour(1*60*60 second
   s or 3600 seconds)
3. setcookie("CookieName", "CookieValue", time()+1*60*60, "/mypath/", "mydomain.com", 1);
```

PHP $_COOKIE

PHP $_COOKIE superglobal variable is used to get cookie.

Example

```php
$value=$_COOKIE["CookieName"];//returns cookie value
```

PHP Cookie Example

File: cookie1.php

```php
<?php
setcookie("user", "Sonoo");
?>
<html>
<body>
<?php
if(!isset($_COOKIE["user"])) {
    echo "Sorry, cookie is not found!";
} else {
    echo "<br/>Cookie Value: " . $_COOKIE["user"];
}
?>
</body>
</html>
```

Output:

```
Sorry, cookie is not found!
```

Firstly cookie is not set. But, if you *refresh* the page, you will see cookie is set now.

Output:

```
Cookie Value: Sonoo
```

PHP Delete Cookie

If you set the expiration date in past, cookie will be deleted.

File: cookie1.php

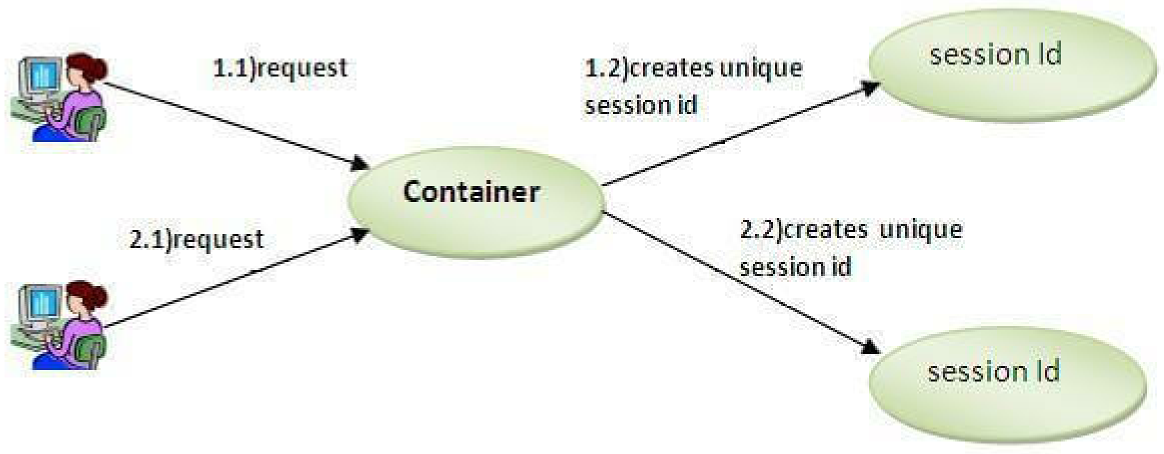
```php
<?php
setcookie ("CookieName", "", time() - 3600);// set the expiration date to one hour ago
?>
```

PHP Session

PHP session is used to store and pass information from one page to another temporarily (until user close the website).

PHP session technique is widely used in shopping websites where we need to store and pass cart information e.g. username, product code, product name, product price etc from one page to another.

PHP session creates unique user id for each browser to recognize the user and avoid conflict between multiple browsers.



PHP session_start() function

PHP session_start() function is used to start the session. It starts a new or resumes existing session. It returns existing session if session is created already. If session is not available, it creates and returns new session.

Syntax

```
1.  bool session_start ( void )
```

Example

```
1.  session_start();
```

PHP $_SESSION

PHP $_SESSION is an associative array that contains all session variables. It is used to set and get session variable values.

Example: Store information

```
1.  $_SESSION["user"] = "Sachin";
```

Example: Get information

```
echo $_SESSION["user"];
```

PHP Session Example

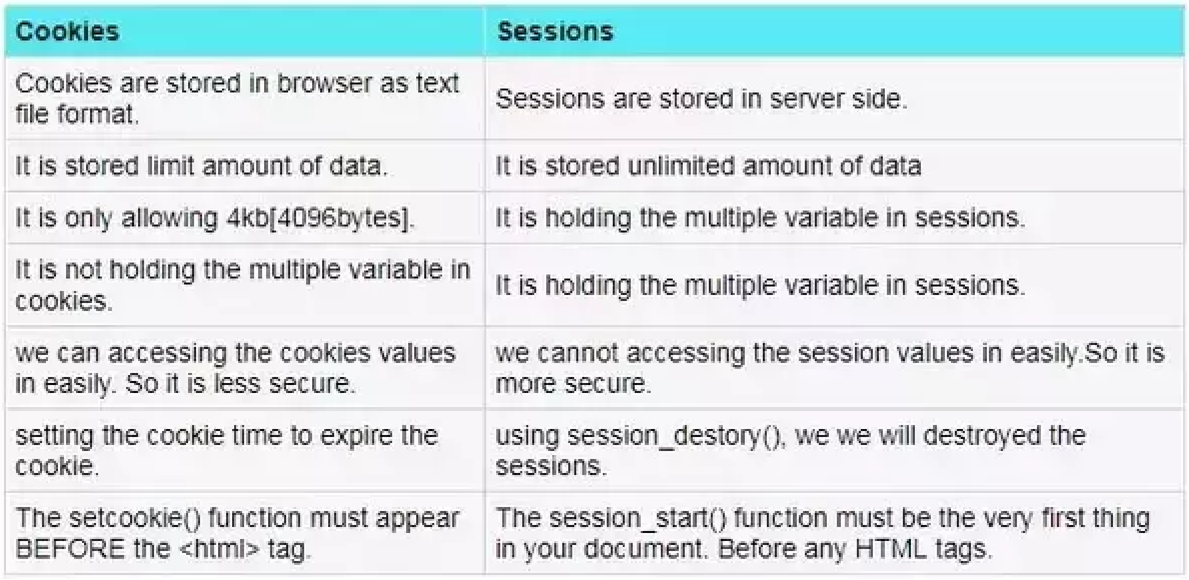
File: session1.php

```
<?php
session_start();
?>
<html>
<body>
<?php
$_SESSION["user"] = "Sachin";
echo "Session information are set successfully.<br/>";
?>
<a href="session2.php">Visit next page</a>
</body>
</html>
```

File: session2.php

```
<?php
session_start();
?>
<html>
<body>
<?php
echo "User is: ".$_SESSION["user"];
?>
</body>
</html>
```

Difference between session and cookies



Cookies	Sessions
Cookies are stored in browser as text file format.	Sessions are stored in server side.
It is stored limit amount of data.	It is stored unlimited amount of data
It is only allowing 4kb[4096bytes].	It is holding the multiple variable in sessions.
It is not holding the multiple variable in cookies.	It is holding the multiple variable in sessions.
we can accessing the cookies values in easily. So it is less secure.	we cannot accessing the session values in easily.So it is more secure.
setting the cookie time to expire the cookie.	using session_destory(), we we will destroyed the sessions.
The setcookie() function must appear BEFORE the <html> tag.	The session_start() function must be the very first thing in your document. Before any HTML tags.

Few Programs:

1.Factorial of a no using form

```
<html>
<head>
<title>Factorial Program using loop in PHP</title>
</head>
<body>
<form method="post" action=<?php echo $_SERVER['PHP_SELF']; ?>>
  Enter the Number:<br>
  <input type="number" name="number" id="number">
  <input type="submit" name="submit" value="Submit" />
</form>
<?php
  if($_POST){
    $fact = 1;
    //getting value from input text box 'number'
    $number = $_POST['number'];
    echo "Factorial of $number:<br><br>";
    //start loop
    for ($i = 1; $i <= $number; $i++){
      $fact = $fact * $i;
      }
      echo $fact . "<br>";
  }

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

2. To calculate sum of 2 no using form(example of isset function)

```
<html>
<head>
<title>PHP isset() example</title>
</head>
<body>

<form method="post" action=<?php echo $_SERVER['PHP_SELF']; ?>>
```

```
Enter value1 :<input type="text" name="str1"><br/>
Enter value2 :<input type="text" name="str2"><br/>
<input type="submit" value="Sum" name="Submit1"><br/><br/>

<?php
if(isset($_POST["Submit1"]))
{
$sum=$_POST["str1"] + $_POST["str2"];
echo "The sum = ". $sum;

}
?>

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

3. To create a registration form having fields like name , address, email, gender and apply validations on email , name etc.

First file: registration.php

```
<html>
<body>
<form method="post" action="validation.php" align="center">
   <label>Name<input type="text" name="name" placeholder="enter your name"></label><br><br>
   <label>Email<input type="text" name="email"></label><br><br>
   Extra Info<br><textarea name="extra_info"></textarea></label><br>
   Gender:
         <input type="radio" name="gender" value="female"> Female
         <input type="radio" name="gender" value="male"> Male
         <br/>



   <input type="submit" name="submit" value="Submit">
</form>
</body>
```

```
</html>
```

Second file: validation.php

```
<?php
 //Setup an empty array.
$errors = array();

//If our form has been submitted.
if(isset($_POST['submit'])){

    //Get the values of our form fields.
    $name = isset($_POST['name']) ? $_POST['name'] : null;
    $email = isset($_POST['email']) ? $_POST['email'] : null;
    $extraInfo = isset($_POST['extra_info']) ? $_POST['extra_info'] : null;

    //Check the name and make sure that it isn't a blank/empty string.
    if(strlen(trim($name)) === 0){
        //Blank string, add error to $errors array.
     echo "You must enter your name!"."<br>";
    }
    else {
        echo $_POST['name']."<br>";  }

    //Make sure that the email address is valid.
    if(!filter_var($email, FILTER_VALIDATE_EMAIL)) {
        //$email is not a valid email. Add error to $errors array.
   echo   "That is not a valid email address!";}
        else
        {
            echo $_POST['email'];
        }

}
?>
```

4. Php script to pass variable between pages using get or post method

First file: index.php

```
<html>
<body>

<form action="registration.php" method="get" align="center">
        <p> <h2>Registration Form</h2></p>
<b>Name: <input type="text" name="name"><br>
<b>Email: <input type="text" name="email"><br>
<b>Gender:Male<input type="radio" name="gender" value="Male">
Female<input type="radio" name="gender" value="Female"><br><br>
<input type="submit">
</form>

</body>
</html>
```

Second File:registration.php

```
<html>
<body>

Welcome <?php echo $_GET["name"]; ?>!<br>
Your email address is <?php echo $_GET["email"]; ?><br>
Gender is <?php echo $_GET["gender"]; ?>

</body>
</html>
```