# PHP File Open/Read/Close

---

In this chapter we will teach you how to open, read, and close a file on the server.

---

## PHP Open File - fopen()

A better method to open files is with the fopen() function. This function gives you more options than the readfile() function.

We will use the text file, "webdictionary.txt", during the lessons:

AJAX = Asynchronous JavaScript and XML
CSS = Cascading Style Sheets
HTML = Hyper Text Markup Language
PHP = PHP Hypertext Preprocessor
SQL = Structured Query Language
SVG = Scalable Vector Graphics
XML = EXtensible Markup Language

The first parameter of fopen() contains the name of the file to be opened and the second parameter specifies in which mode the file should be opened. The following example also generates a message if the fopen() function is unable to open the specified file:

Example

```
<?php
$myfile = fopen("webdictionary.txt", "r") or die("Unable to open file!");
echo fread($myfile,filesize("webdictionary.txt"));
fclose($myfile);
?>
```

**Tip:** The fread() and the fclose() functions will be explained below.

The file may be opened in one of the following modes:

| Modes | Description |
| --- | --- |
| r | **Open a file for read only**. File pointer starts at the beginning of the file |
| w | **Open a file for write only**. Erases the contents of the file or creates a new file if it doesn't exist. File pointer starts at the beginning of the file |
| a | **Open a file for write only**. The existing data in file is preserved. File pointer starts at the end of the file. Creates a new file if the file doesn't exist |
| x | **Creates a new file for write only**. Returns FALSE and an error if file already exists |
| r+ | **Open a file for read/write**. File pointer starts at the beginning of the file |
| w+ | **Open a file for read/write**. Erases the contents of the file or creates a new file if it doesn't exist. File pointer starts at the beginning of the file |
| a+ | **Open a file for read/write**. The existing data in file is preserved. File pointer starts at the end of the file. Creates a new file if the file doesn't exist |

| | |
|---|---|
| x+ | **Creates a new file for read/write**. Returns FALSE and an error if file already exists |

---

---

**PHP Read File - fread()**

The fread() function reads from an open file.

The first parameter of fread() contains the name of the file to read from and the second parameter specifies the maximum number of bytes to read.

The following PHP code reads the "webdictionary.txt" file to the end:

```
fread($myfile,filesize("webdictionary.txt"));
```

---

**PHP Close File - fclose()**

The fclose() function is used to close an open file.

It's a good programming practice to close all files after you have finished with them. You don't want an open file running around on your server taking up resources!

The fclose() requires the name of the file (or a variable that holds the filename) we want to close:

```
<?php
$myfile = fopen("webdictionary.txt", "r");
// some code to be executed....
fclose($myfile);
?>
```

---

**PHP Read Single Line - fgets()**

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

The example below outputs the first line of the "webdictionary.txt" file:

Example

```
<?php
$myfile = fopen("webdictionary.txt", "r") or die("Unable to open file!");
echo fgets($myfile);
fclose($myfile);
?>
```

**Note:** After a call to the fgets() function, the file pointer has moved to the next line.

---

## PHP Check End-Of-File - feof()

The feof() function checks if the "end-of-file" (EOF) has been reached.

The feof() function is useful for looping through data of unknown length.

The example below reads the "webdictionary.txt" file line by line, until end-of-file is reached:

Example

```
<?php
$myfile = fopen("webdictionary.txt", "r") or die("Unable to open file!");
// Output one line until end-of-file
while(!feof($myfile)) {
  echo fgets($myfile) . "<br>";
}
fclose($myfile);
?>
```

---

## PHP Read Single Character - fgetc()

The fgetc() function is used to read a single character from a file.

The example below reads the "webdictionary.txt" file character by character, until end-of-file is reached:

Example

```
<?php
$myfile = fopen("webdictionary.txt", "r") or die("Unable to open file!");
// Output one character until end-of-file
while(!feof($myfile)) {
  echo fgetc($myfile);
}
fclose($myfile);
?>
```

**Note:** After a call to the fgetc() function, the file pointer moves to the next character.

**PHP Write to File - fwrite()**

The fwrite() function is used to write to a file.

The first parameter of fwrite() contains the name of the file to write to and the second parameter is the string to be written.

The example below writes a couple of names into a new file called "newfile.txt":

Example

```
<?php
$myfile = fopen("newfile.txt", "w") or die("Unable to open file!");
$txt = "John Doe\n";
fwrite($myfile, $txt);
$txt = "Jane Doe\n";
fwrite($myfile, $txt);
fclose($myfile);
?>
```

Notice that we wrote to the file "newfile.txt" twice. Each time we wrote to the file we sent the string $txt that first contained "John Doe" and second contained "Jane Doe". After we finished writing, we closed the file using the fclose() function.

If we open the "newfile.txt" file it would look like this:

John Doe
Jane Doe

## PHP File Formats Support

PHP file functions support a wide range of file formats that include;

- File.txt
- File.log
- File.custom_extension i.e. file.xyz
- File.csv
- File.gif, file.jpg etc
- Files provide a permanent cost effective data storage solution for simple data compared to databases that require other software and skills to manage DBMS systems.
- You want to store simple data such as server logs for later retrieval and analysis
- You want to store program settings i.e. program.ini

## PHP files Functions

PHP provides a convenient way of working with files via its rich collection of built in functions.

Operating systems such as Windows and MAC OS are not case sensitive while Linux or Unix operating systems are case sensitive.

Adopting a naming conversion such as lower case letters only for file naming is a good practice that ensures maximum cross platform compatibility.

Let's now look at some of the most commonly used PHP file functions.

## PHP File_exists Function

This function is used to determine whether a file exists or not.

- It comes in handy when we want to know if a file exists or not before processing it.
- You can also use this function when creating a new file and you want to ensure that the file does not already exist on the server.

The file_exist function has the following syntax.

```
<?php
file_exists($filename);
?>
```

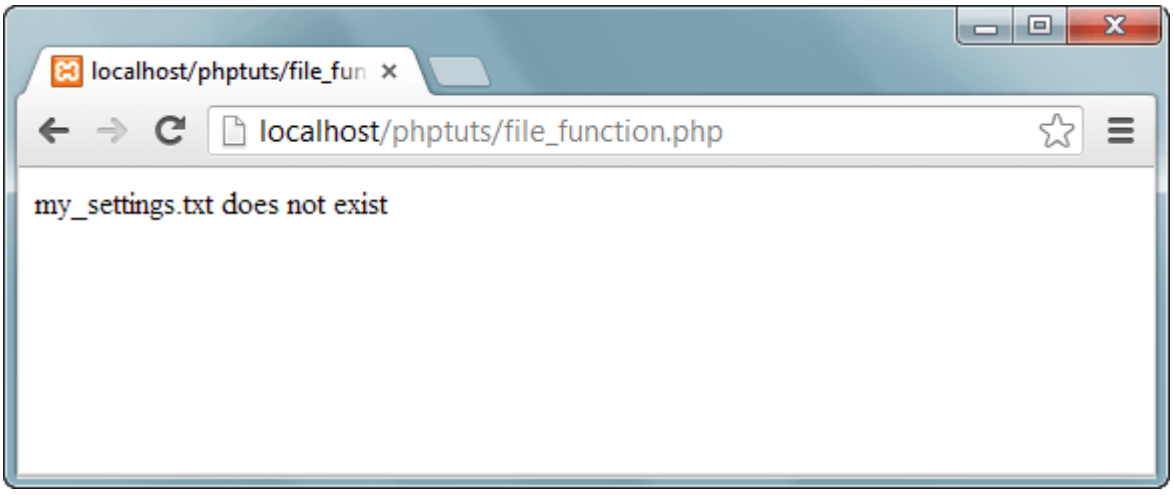
HERE,

- “file_exists()” is the PHP function that returns true if the file exists and false if it does not exist.
- “$file_name” is the path and name of the file to be checked

The code below uses file_exists function to determine if the file my_settings.txt exists.

```
<?php
if (file_exists('my_settings.txt'))
 {
      echo 'file found!';
 }
else
 {
      echo 'my_settings.txt does not exist';
  }
?>
```

Save the above code in a file named file_function.php Assuming you saved the file in phptuts folder in htdocs, open the URL **http://localhost/phptuts/file_function.php** in your browser You will get the following results.




**PHP Fopen Function**

The fopen function is used to open files. It has the following syntax

```
<?php
```

```
fopen($file_name,$mode,$use_include_path,$context);
 ?>
```

HERE,

- “fopen” is the PHP open file function
- “$file_name” is the name of the file to be opened
- “$mode” is the mode in which the file should be opened, the table below shows the modes

| **Mode** | **Description** |
|---|---|
| r | • Read file from beginning.<br>• Returns false if the file doesn’t exist.<br>• Read only |
| r+ | • Read file from beginning<br>• Returns false if the file doesn’t exist.<br>• Read and write |
| w | • Write to file at beginning<br>• truncate file to zero length<br>• If the file doesn’t exist attempt to create it.<br>• Write only |
| w+ | • Write to file at beginning, truncate file to zero length<br>• If the file doesn’t exist attempt to create it.<br>• Read and Write |
| a | • Append to file at end<br>• If the file doesn’t exist attempt to create it.<br>• Write only |
| a+ | • Php append to file at end<br>• If the file doesn’t exist attempt to create it<br>• Read and write |

- “$use_include_path” is optional, default is false, if set to true, the function searches in the include path too.
- “$context” is optional, can be used to specify the context support.

**PHP Fwrite Function**

The fwrite function is used to write files.

It has the following syntax

```
<?php
fwrite($handle, $string, $length);
?>
```

HERE,

- “fwrite” is the PHP function for writing to files
- “$handle” is the file pointer resource
- “$string” is the data to be written in the file.
- “$length” is optional, can be used to specify the maximum file length.

**PHP Fclose Function**

Is is used to close a file in php which is already open

It has the following syntax.

```
<?php
fclose($handle);
?>
```

HERE,

- “fclose” is the PHP function for closing an open file
- “$handle” is the file pointer resource.

Let’s now look at an example that creates my_settings.txt.

We will use the following functions.

- Fopen
- Fwrite
- fclose

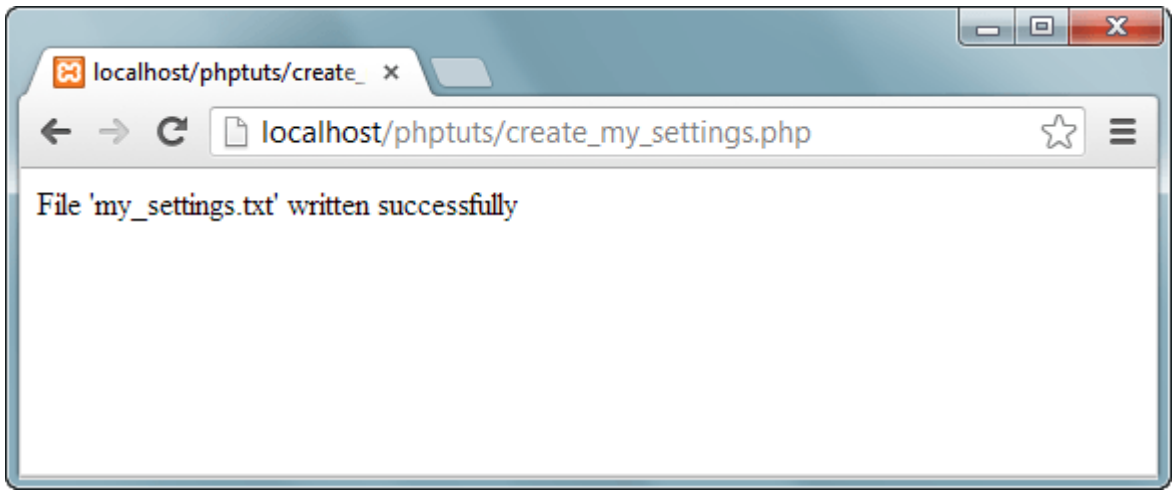
The code below “create_my_settings_file.php” implements the above example.

| | |
|---|---|
| Open a file | <?php<br>$fh = fopen("my_settings.txt", 'w')<br>or<br>die("Failed to create file"); ?> |
| Closing a file | <?php<br>fclose($fh);<br>?> |
| Create File | <?php<br>$fh = fopen("my_settings.txt", 'w') or die("Failed to create file");<br>$text = <<<_END<br>localhost;root;pwd1234;my_database<br>_END;<br>fwrite($fh, $text) or die("Could not write to file");<br>fclose($fh);<br>echo "File 'my_settings.txt' written successfully"; ?> |

**Testing the code**

Open the URL **http://localhost/phptuts/create_my_settings.php** in your browser.

You will get the following page




Note: if your disk is full or you do not have permission to write files, you will get an error message.

Switch back to the URL **http://localhost/phptuts/file_function.php** .

What results do you get?

**PHP Fgets Function**

The fgets function is used to read php files line by line. It has the following basic syntax. fgets($handle); HERE,

- "$fgets" is the PHP function for reading file lines
- "$handle" is the file pointer resource.

Let's now look at an example that reads my_settings.txt file using the fopen and fgets functions.

The code below read_my_settings.php implements the above example.

```
<?php
$fh = fopen("my_settings.txt", 'r') or die("File does not exist or you lack permiss
ion to open it");
$line = fgets($fh);
echo $line; fclose($fh);
 ?>
```

HERE,

- "fopen" function returns the pointer to the file specified in the file path
- "die()" function is called if an error occurs. It displays a message and exists execution of the script

## PHP Copy Function

The PHP copy function is used to copy files. It has the following basic syntax. copy($file,$copied_file); HERE,

- “$file” specifies the file path and name of the file to be copied.
- “copied_file” specified the path and name of the copied file

The code below illustrates the implementation

```
<?php
copy('my_settings.txt', 'my_settings_backup.txt') or die("Could not copy file");
echo "File successfully copied to 'my_settings_backup.txt'";
?>
```

## Deleting a file

The unlink function is used to delete the file. The code below illustrates the implementation.

```
<?php
if (!unlink('my_settings_backup.txt'))
{
    echo "Could not delete file";
 }
else
{
    echo "File 'my_settings_backup.txt' successfully deleted";
}
 ?>
```

## PHP File_get_contents Function

The file_get_contents function is used to read the entire file contents.

The code below illustrates the implementation.

The difference between file_get_contents and fgets is that file_get_contents returns the file data as a string while fgets reads the file line by line.

```
<?php
echo "<pre>"; // Enables display of line feeds
echo file_get_contents("my_settings.txt");
echo "</pre>"; // Terminates pre tag
```

```
?>
```

**Summary**

- A file is a resource for storing data
- PHP has a rich collection of built in functions that simplify working with files.
- Common file functions include fopen, fclose, file_get_contents
- The table below shows a summary of the functions covered

| Function | Description |
|---|---|
| File_exists | Used to determine if a file exists or not |
| fopen | Used to open a file. Returns a pointer to the opened file |
| fwrite | Used to write to files |
| fclose | Used to open closed files |
| fgets | Used to read a file line by line |
| copy | Used to copy an existing file |
| unlink | Used to delete an existing file |
| file_get_contents | Used to return the contents of a file as a string |

**PHP Sessions**

A session is a way to store information (in variables) to be used across multiple pages.

Unlike a cookie, the information is not stored on the users computer.

---

What is a PHP Session?

When you work with an application, you open it, do some changes, and then you close it. This is much like a Session. The computer knows who you are. It knows when you start the application and when you end. But on the internet there is one problem: the web server does not know who you are or what you do, because the HTTP address doesn't maintain state.

Session variables solve this problem by storing user information to be used across multiple pages (e.g. username, favorite color, etc). By default, session variables last until the user closes the browser.

So; Session variables hold information about one single user, and are available to all pages in one application.

---

**Start a PHP Session**

A session is started with the session_start() function.

Session variables are set with the PHP global variable: $_SESSION.

Now, let's create a new page called "demo_session1.php". In this page, we start a new PHP session and set some session variables:

Example

```
<?php
// Start the session
session_start();
?>
<!DOCTYPE html>
<html>
<body>

<?php
// Set session variables
$_SESSION["favcolor"] = "green";
$_SESSION["favanimal"] = "cat";
echo "Session variables are set.";
?>
</body>
</html>
```

## Get PHP Session Variable Values

Next, we create another page called "demo_session2.php". From this page, we will access the session information we set on the first page ("demo_session1.php").

Notice that session variables are not passed individually to each new page, instead they are retrieved from the session we open at the beginning of each page (session_start()).

Also notice that all session variable values are stored in the global $_SESSION variable:

Example

```
<?php
session_start();
?>
<!DOCTYPE html>
<html>
<body>

<?php
// Echo session variables that were set on previous page
echo "Favorite color is " . $_SESSION["favcolor"] . ".<br>";
echo "Favorite animal is " . $_SESSION["favanimal"] . ".";
?>

</body>
</html>
```

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

Example

```
<?php
session_start();
?>
<!DOCTYPE html>
```

```
<html>
<body>

<?php
print_r($_SESSION);
?>

</body>
</html>
```

**Modify a PHP Session Variable**

To change a session variable, just overwrite it:

Example

```
<?php
session_start();
?>
<!DOCTYPE html>
<html>
<body>

<?php
// to change a session variable, just overwrite it
$_SESSION["favcolor"] = "yellow";
print_r($_SESSION);
?>

</body>
</html>
```

---

**Destroy a PHP Session**

To remove all global session variables and destroy the session,
use session_unset() and session_destroy():

Example

```
<?php
session_start();
?>
<!DOCTYPE html>
<html>
<body>

<?php
// remove all session variables
session_unset();

// destroy the session
session_destroy();
?>

</body>
</html>
```

**PHP setcookie() Function**

Example

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

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

```
<!DOCTYPE html>
<?php
$cookie_name = "user";
$cookie_value = "John Doe";
setcookie($cookie_name, $cookie_value, time() + (86400 * 30), "/"); // 86400 =
1 day
?>
<html>
<body>
```

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

</body>
</html>
```

---

**Definition and Usage**

The setcookie() function defines a cookie to be sent along with the rest of the HTTP headers.

A cookie is often used to identify a user. A cookie is a small file that the server embeds on the user's computer. Each time the same computer requests a page with a browser, it will send the cookie too. With PHP, you can both create and retrieve cookie values.

The name of the cookie is automatically assigned to a variable of the same name. For example, if a cookie was sent with the name "user", a variable is automatically created called $user, containing the cookie value.

**Note:** The setcookie() function must appear BEFORE the <html> tag.

**Note:** The value of the cookie is automatically URLencoded when sending the cookie, and automatically decoded when received (to prevent URLencoding, use setrawcookie() instead).

Syntax

setcookie(*name, value, expire, path, domain, secure, httponly*);

Parameter Values

| Parameter | Description |
| --- | --- |
| *name* | Required. Specifies the name of the cookie |
| *value* | Optional. Specifies the value of the cookie |
| *expire* | Optional. Specifies when the cookie expires. The value: time()+86400*30, will set the cookie to expire in 30 days. If this parameter is omitted or set to 0, the cookie will expire at the end of the session (when the browser closes). Default is 0 |
| *path* | Optional. Specifies the server path of the cookie. If set to "/", the cookie will be available within the entire domain. If set to "/php/", the cookie will only be available within the php directory and all sub-directories of php. The default value is the current directory that the cookie is being set in |
| *domain* | Optional. Specifies the domain name of the cookie. To make the cookie available on all subdomains of example.com, set domain to "example.com". Setting it to www.example.com will make the cookie only available in the www subdomain |
| *secure* | Optional. Specifies whether or not the cookie should only be transmitted over a secure HTTPS connection. TRUE indicates that the cookie will only be set if a secure |

| | |
|---|---|
| | connection exists. Default is FALSE |
| *httponly* | Optional. If set to TRUE the cookie will be accessible only through the HTTP protocol (the cookie will not be accessible by scripting languages). This setting can help to reduce identity theft through XSS attacks. Default is FALSE |

---

---

**Technical Details**

| | |
|---|---|
| **Return Value:** | TRUE on success. FALSE on failure |
| **PHP Version:** | 4+ |
| **PHP Changelog:** | PHP 5.5 - A Max-Age attribute was included in the Set-Cookie header sent to the client<br>PHP 5.2 - The httponly parameter was added |

---

More Examples

Example

Several expire dates for cookies:

```
<?php
$value = "Hello world!";
```

```
// cookie will expire when the browser close
setcookie("myCookie", $value);

// cookie will expire in 1 hour
setcookie("myCookie", $value, time() + 3600);

// cookie will expire in 1 hour, and will only be available
// within the php directory + all sub-directories of php
setcookie("myCookie", $value, time() + 3600, "/php/");
?>
<html>
<body>

...some code...

</body>
</html>
```

Example

To modify a cookie, just set (again) the cookie using the setcookie() function:

```
<?php
$cookie_name = "user";
$cookie_value = "Alex Porter";
setcookie($cookie_name, $cookie_value, time() + (86400 * 30), "/");
?>
<html>
<body>

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

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

Example

To delete a cookie, use the setcookie() function with an expiration date in the past:

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

<?php
echo "Cookie 'user' is deleted.";
?>

</body>
</html>
```

Example

Create a small script that checks whether cookies are enabled. First, try to create a test cookie with the setcookie() function, then count the $_COOKIE array variable:

```
<?php
setcookie("test_cookie", "test", time() + 3600, '/');
?>
<html>
<body>

<?php
if(count($_COOKIE) > 0) {
    echo "Cookies are enabled.";
} else {
    echo "Cookies are disabled.";
}
?>
```

</body>
</html>

# PHP Cookies

### What is a Cookie?

A cookie is often used to identify a user. A cookie is a small file that the server embeds on the user's computer. Each time the same computer requests a page with a browser, it will send the cookie too. With PHP, you can both create and retrieve cookie values.

---

### Create Cookies With PHP

A cookie is created with the setcookie() function.

Syntax

setcookie(*name, value, expire, path, domain, secure, httponly*);

Only the *name* parameter is required. All other parameters are optional.

---

### PHP Create/Retrieve a Cookie

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

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

Example

```
<?php
$cookie_name = "user";
$cookie_value = "John Doe";
setcookie($cookie_name, $cookie_value, time() + (86400 * 30), "/"); // 86400 =
1 day
?>
```

```
<html>
<body>

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

</body>
</html>
```

**Note:** The value of the cookie is automatically URLencoded when sending the cookie, and automatically decoded when received (to prevent URLencoding, use setrawcookie() instead).

---

---

**Modify a Cookie Value**

To modify a cookie, just set (again) the cookie using the setcookie() function:

Example

```
<?php
$cookie_name = "user";
$cookie_value = "Alex Porter";
setcookie($cookie_name, $cookie_value, time() + (86400 * 30), "/");
?>
<html>
<body>

<?php
if(!isset($_COOKIE[$cookie_name])) {
  echo "Cookie named '" . $cookie_name . "' is not set!";
```

```
} else {
  echo "Cookie '" . $cookie_name . "' is set!<br>";
  echo "Value is: " . $_COOKIE[$cookie_name];
}
?>

</body>
</html>
```

---

**Delete a Cookie**

To delete a cookie, use the setcookie() function with an expiration date in the past:

Example

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

<?php
echo "Cookie 'user' is deleted.";
?>

</body>
</html>
```

---

**Check if Cookies are Enabled**

The following example creates a small script that checks whether cookies are enabled. First, try to create a test cookie with the setcookie() function, then count the $_COOKIE array variable:

Example

```
<?php
setcookie("test_cookie", "test", time() + 3600, '/');
?>
<html>
<body>

<?php
if(count($_COOKIE) > 0) {
  echo "Cookies are enabled.";
} else {
  echo "Cookies are disabled.";
}
?>

</body>
</html>
```

**Remove a cookie using PHP**

**Cookie:** A Cookie is a small file sent by the server to preserve stateful information for a user. It is stored on the client's computer and sent to the server every time the user makes a request for the same page.
To create cookies you can set the cookie by using the **setcookie()** function of the PHP.
**Syntax:**
setcookie(name, value, expire, path, domain, secure, httponly)

**Parameters:** This function accepts seven parameters as mentioned above and described below:

- **name:** The name of the cookie.
- **value:** The value you want to store in the cookie.
- **expire-time:** It is the number of seconds until the cookie will be kept on the user's machine by the browser. After that, it will automatically be deleted. If not set then the cookie will be preserved by browser until it is open.
- **path:** It determines for which directories cookie will valid. If you want to access it in all directories then put it on "/", i.e. the cookie is accessible in the entire domain. Otherwise, the cookie will be limited to the subdirectory.
- **domain:** It is used to define the access hierarchy for the cookie. For example, if you set this to "yourdomain.com", it will be accessible via all the

subdomains also. but if it set to “sub.yourdomain.com”, it will be accessible by “sub.yourdomain.com” and its subdomains.

- **secure:** It determines how the cookie will be sent, via HTTP or HTTPS. If set to true then the cookie will be sent via HTTPS only, otherwise, it will be sent via HTTP. Its default value is **false**.
- **httponly:** If it set to true, the cookie is accessible only either via HTTP or HTTPS. That means the client code (like Javascript) can not access the cookie.

Out of the above parameters, only the **first two** parameters are mendatory. Others are optional parameters. If you want to preserve the cookie, then provide the *expire-time* parameter.

**Note:** It is stored in the global array named **$_COOKIE**.
**Creating a Cookie:** As mentioned earlier, we can set cookies by using the function **setcookie()**.

- **Example:**

```
<!DOCTYPE html>



<?php

$cookie_name = "gfg";

$cookie_value = "GeeksforGeeks";



// 86400 = 1 day

setcookie($cookie_name, $cookie_value, time() +
(86400 * 15), "/");

?>
```

```
<html>

<body>

   <?php

   if(!isset($_COOKIE[$cookie_name])) {

       echo "Cookie named '" . $cookie_name . "' is not
set!";

   }

   else {

       echo "Cookie '" . $cookie_name . "' is set!<br>";

       echo "Value is: " . $_COOKIE[$cookie_name];

   }

   ?>



</body>



</html>
```

- **Output:**
- Cookie 'gfg' is set!
- Value is: GeeksforGeeks

**Deleting Cookie:** There is no special dedicated function provided in PHP to delete a cookie. All we have to do is to update the **expire-time** value of the cookie by setting it to a past time using the **setcookie()** function. A very simple way of doing this is to deduct a few seconds from the current time.

- **Syntax:**
- setcookie(name, time() - 3600);
- **Example:**

```
<!DOCTYPE html>

<?php



// Set the expiration date to one hour ago

setcookie("gfg", "", time() - 3600);

?>



<html>



<body>



    <?php

    echo "Cookie 'gfg' is deleted.";

    ?>



</body>
```

```
</html>
```

- **Output:**
- Cookie 'gfg' is deleted.

**Note:** The setcookie() function must appear before the <html> tag.

## PHP Date and Time

The PHP date() function is used to format a date and/or a time.

---

The PHP Date() Function

The PHP date() function formats a timestamp to a more readable date and time.

Syntax

date(*format*,*timestamp*)

| Parameter | Description |
|---|---|
| format | Required. Specifies the format of the timestamp |
| timestamp | Optional. Specifies a timestamp. Default is the current date and time |

A timestamp is a sequence of characters, denoting the date and/or time at which a certain event occurred.

**Get a Date**

The required *format* parameter of the date() function specifies how to format the date (or time).

Here are some characters that are commonly used for dates:

- d - Represents the day of the month (01 to 31)

- m - Represents a month (01 to 12)
- Y - Represents a year (in four digits)
- l (lowercase 'L') - Represents the day of the week

Other characters, like"/", ".", or "-" can also be inserted between the characters to add additional formatting.

The example below formats today's date in three different ways:

Example

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

**PHP Tip - Automatic Copyright Year**

Use the date() function to automatically update the copyright year on your website:

Example

```
&copy; 2010-<?php echo date("Y");?>
```

Get a Time

Here are some characters that are commonly used for times:

- H - 24-hour format of an hour (00 to 23)
- h - 12-hour format of an hour with leading zeros (01 to 12)
- i - Minutes with leading zeros (00 to 59)
- s - Seconds with leading zeros (00 to 59)
- a - Lowercase Ante meridiem and Post meridiem (am or pm)

The example below outputs the current time in the specified format:

Example

```
<?php
echo "The time is " . date("h:i:sa");
?>
```

Note that the PHP date() function will return the current date/time of the server!

**Get Your Time Zone**

If the time you got back from the code is not correct, it's probably because your server is in another country or set up for a different timezone.

So, if you need the time to be correct according to a specific location, you can set the timezone you want to use.

The example below sets the timezone to "America/New_York", then outputs the current time in the specified format:

Example

```
<?php
date_default_timezone_set("America/New_York");
echo "The time is " . date("h:i:sa");
?>
```

**Create a Date With mktime()**

The optional *timestamp* parameter in the date() function specifies a timestamp. If omitted, the current date and time will be used (as in the examples above).

The PHP mktime() function returns the Unix timestamp for a date. The Unix timestamp contains the number of seconds between the Unix Epoch (January 1 1970 00:00:00 GMT) and the time specified.

Syntax

mktime(*hour, minute, second, month, day, year*)

The example below creates a date and time with the date() function from a number of parameters in the mktime() function:

Example

```
<?php
$d=mktime(11, 14, 54, 8, 12, 2014);
echo "Created date is " . date("Y-m-d h:i:sa", $d);
?>
```

Create a Date From a String With strtotime()

The PHP strtotime() function is used to convert a human readable date string into a Unix timestamp (the number of seconds since January 1 1970 00:00:00 GMT).

**Syntax**

strtotime(*time, now*)

The example below creates a date and time from the strtotime() function:

Example

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

PHP is quite clever about converting a string to a date, so you can put in various values:

Example

```
<?php
$d=strtotime("tomorrow");
echo date("Y-m-d h:i:sa", $d) . "<br>";

$d=strtotime("next Saturday");
echo date("Y-m-d h:i:sa", $d) . "<br>";

$d=strtotime("+3 Months");
echo date("Y-m-d h:i:sa", $d) . "<br>";
?>
```

However, strtotime() is not perfect, so remember to check the strings you put in there.

---

**More Date Examples**

The example below outputs the dates for the next six Saturdays:

Example

```
<?php
$startdate = strtotime("Saturday");
```

```
$enddate = strtotime("+6 weeks", $startdate);

while ($startdate < $enddate) {
  echo date("M d", $startdate) . "<br>";
  $startdate = strtotime("+1 week", $startdate);
}
?>
```

The example below outputs the number of days until 4th of July:

**Example**

```
<?php
$d1=strtotime("July 04");
$d2=ceil(($d1-time())/60/60/24);
echo "There are " . $d2 ." days until 4th of July.";
?>
```