

PHP

Prithviraj Mohanty
Asst.Prof. Dept of CSIT





The PHP Date() Function

- The PHP date() function is used to format a date and/or a time.
- The PHP date() function formats a timestamp to a more readable date and time.

Syntax

date(format,timestamp)

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

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

Get a Simple Date

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

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.



Example

```
<html>
<body>
<?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");
?>
</body>
</html>
```

OUTPUT:

Today is 2017/12/09

Today is 2017.12.09

Today is 2017-12-09

Today is Saturday.

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 Simple Time

- Characters that are commonly used for times:
 - 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)
- **Note** that the PHP `date()` function will return the current date/time of the server!

Example:

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

OUTPUT:

The time is 11:26:06am



Get Your Time Zone

- If the time you got back from the code is not the right time, 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 a timezone 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 PHP mktime()

The optional timestamp parameter in the date() function specifies a timestamp. If you do not specify a timestamp, the current date and time will be used (as shown in the example).

The 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)



Example:

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

Output:

Created date is 2014-08-12
11:14:54am

Create a Date From a String With PHP strtotime()

The PHP strtotime() function is used to convert a human readable string to a Unix time.

Syntax: *strtotime(time, now);*

Example

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

Output: Created date is 2014-04-15
10:30:00pm



Example:

```
<html> <body>
<?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>";
?>
</body> </html>
```

OUTPUT:

```
2016-11-02 12:00:00am
2016-11-05 12:00:00am
2017-02-01 12:42:09am
```



PHP: Include Files

- The include (or require) statement takes all the text/code/markup that exists in the specified file and copies it into the file that uses the include statement.
- Including files is very useful when you want to include the same PHP, HTML, or text on multiple pages of a website.

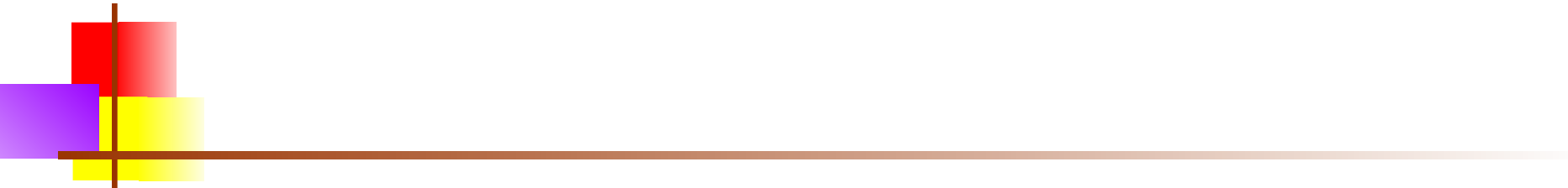
PHP include and require Statements

- It is possible to insert the content of one PHP file into another PHP file (before the server executes it), with the include or require statement.

The include and require statements are identical, except upon failure:

require will produce a fatal error (E_COMPILE_ERROR) and stop the script

include will only produce a warning (E_WARNING) and the script will continue.

- 
- So, if you want the execution to go on and show users the output, even if the include file is missing, use the **include statement**.
 - Otherwise, in case of Framework, CMS, or a complex PHP application coding, always use the require statement to include a key file to the flow of execution. This will help avoid compromising your application's security and integrity, just in-case one key file is accidentally missing.
 - Including files saves a lot of work. This means that you can create a standard header, footer, or menu file for all your web pages. Then, when the header needs to be updated, you can only update the header include file.

Syntax

include 'filename';

or

require 'filename';



PHP include Examples

- Assume we have a standard footer file called "**footer.php**", that looks like this:

```
<?php  
echo "<p>Copyright &copy; 1999-" .  
    date("Y") . " iter.ac.in</p>";?>
```

To include the footer file in a page,
use the include statement:

Example-1

```
<html> <body>  
<h1>Welcome to my home pag</h1>  
<p>Some text.</p>  
<p>Some more text.</p>  
<?php include 'footer.php';?>  
</body> </html>
```

Output:

Welcome to my home
page!

Some text.

Some more text.

Copyright © 1999-2017 iter.ac.in



Example-2

- Assume we have a standard menu file called "**menu.php**":

```
<?php  
echo  
    '<a href="/default.asp">Home</a>  
    <a href="/html/default.asp">HTML  
    Tutorial</a>  
    <a href="/css/default.asp">CSS  
    Tutorial</a>  
    <a href="/js/default.asp">JavaScript  
    Tutorial</a>  
    <a href="default.asp">PHP  
    Tutorial</a>';  
?>
```

Example

```
<html>  
<body>  
    <div class="menu">  
        <?php include 'menu.php';?>  
    </div>  
    <h1>Welcome to my home page!  
        </h1>  
    <p>Some text.</p>  
    <p>Some more text.</p>  
</body>  
</html>
```



Example 3

- Assume we have a file called "**vars.php**", with some variables defined:

```
<?php  
$color='red';  
$car='BMW';?>
```

- Then, if we include the "vars.php" file, the variables can be used in the calling file:

Example

```
<html> <body>  
<h1>Welcome to my home page!/h1>  
<?php include 'vars.php';  
echo "I have a $color $car.";  
?> </body> </html>
```

PHP include vs. require

The require statement is also used to include a file into the PHP code.

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



Example(include)

```
<html><body>
<h1>Welcome to my home page!
    </h1>
<?php include 'noFileExists.php';
echo "I have a $color $car.";
?>
</body>
</html>
```

Example(require)

```
<html><body>
<h1>Welcome to my home page!</h1>
<?php require 'noFileExists.php';
echo "I have a $color $car.";
?>
</body>
</html>
```

If we do the same example using the require statement, the echo statement will not be executed because the script execution dies after the require statement returned a **fatal error**:

- Use **require** when the file is required by the application.
- Use **include** when the file is not required and application should continue when file is not found.



PHP: File Handling

- File handling is an important part of any web application. You often need to open and process a file for different tasks.

PHP Manipulating Files

- PHP has several functions for creating, reading, uploading, and editing files.
- When you are manipulating files you must be very careful.
- You can do a lot of damage if you do something wrong.
- **Common errors are:** editing the wrong file, filling a hard-drive with garbage data, and deleting the content of a file by accident.

PHP readfile() Function

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

Assume we have a text file called "**webdictionary.txt**", stored on the server, that looks like this:

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 PHP code to read the file and write it to the output buffer is as follows (the readfile() function returns the number of bytes read on success):

```
<?php echo readfile("webdictionary.txt");?>
```



PHP: File Open/Read/Close

- 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", containing:

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.

Example:

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

```
<?php
```

```
$myfile =
```

```
    fopen("webdictionary.txt", "r")  
    or die("Unable to open file!");
```

```
echo
```

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

```
fclose($myfile);
```

```
?>
```

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



File Opening Modes:

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

- 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.

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);
?>
```

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.



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: File Create/Write

PHP Create File - fopen()

- The fopen() function is also used to create a file. Maybe a little confusing, but in PHP, a file is created using the same function used to open files.
- If you use fopen() on a file that does not exist, it will create it, given that the file is opened for writing (w) or appending (a).
- The example below creates a new file called "testfile.txt". The file will be created in the same directory where the PHP code resides:

Example

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

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);
```

```
?>
```



PHP Overwriting

- Now that "newfile.txt" contains some data we can show what happens when we open an existing file for writing. All the existing data will be ERASED and we start with an empty file.
- In the example below we open our existing file "newfile.txt", and write some new data into it:

Example

```
<?php
```

```
$myfile = fopen("newfile.txt", "w") or die("Unable to open file!");
```

```
$txt = "Mickey Mouse\n";
```

```
fwrite($myfile, $txt);
```

```
$txt = "Minnie Mouse\n";
```

```
fwrite($myfile, $txt);
```

```
fclose($myfile);
```

```
?>
```



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:
- **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).



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>
```



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>
```



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.
- Tip:** If you need a permanent storage, you may want to store the data in a database.



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

Note: The `session_start()` function must be the very first thing in your document.

Before any HTML tags.



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();
?>
<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>
```



Modify a PHP Session Variable

- To change a session variable, just overwrite it:

Example

```
<?php
session_start();
?>
<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();
?>
<html> <body>
<?php
// remove all session variables
session_unset();
// destroy the session
session_destroy();
?> </body> </html>
```




PHP Filters

- Validating data = Determine if the data is in proper form.
- Sanitizing data = Remove any illegal character from the data.

The PHP Filter Extension

- PHP filters are used to validate and sanitize external input.
- The PHP filter extension has many of the functions needed for checking user input, and is designed to make data validation easier and quicker.

Why Use Filters?

- Many web applications receive external input. External input/data can be:
 - User input from a form
 - Cookies
 - Web services data
 - Server variables
 - Database query results
- **Note:** Invalid submitted data can lead to security problems and break your webpage!
- By using PHP filters you can be sure your application gets the correct input



PHP filter_var() Function

- The filter_var() function both validate and sanitize data.
- The filter_var() function filters a single variable with a specified filter. It takes two pieces of data:
 - The variable you want to check
 - The type of check to use

Sanitize a String

- The following example uses the filter_var() function to remove all HTML tags from a string:

Example

```
<?php
$str = "<h1>Hello World!</h1>";
$newstr = filter_var($str,
    FILTER_SANITIZE_STRING);
echo $newstr;
?>
```



Validate an Integer

- The following example uses the `filter_var()` function to check if the variable `$int` is an integer.

Example-1

```
<?php
$int = 100;
if (!filter_var($int,
    FILTER_VALIDATE_INT) ===
    false) {
    echo("Integer is valid");
} else {
    echo("Integer is not valid");
}
?>
```

Example-2

```
<?php
$int = 0;
if (filter_var($int,
    FILTER_VALIDATE_INT) === 0
    || !filter_var($int,
    FILTER_VALIDATE_INT) ===
    false) {
    echo("Integer is valid");
} else {
    echo("Integer is not valid");
}
?>
```



Validate an IP Address

- The following example uses the `filter_var()` function to check if the variable `$ip` is a valid IP address:

Example

```
<?php
$ip = "127.0.0.1";

if (!filter_var($ip,
    FILTER_VALIDATE_IP) === false) {
    echo("$ip is a valid IP address");
} else {
    echo("$ip is not a valid IP address");
}
?>
```

Sanitize and Validate an Email Address

The following example uses the `filter_var()` function to first remove all illegal characters from the `$email` variable, then check if it is a valid email address:



Example

```
<?php
```

```
$email = "raj.iter@example.com";
```

```
// Remove all illegal characters from email
```

```
$email = filter_var($email, FILTER_SANITIZE_EMAIL);
```

```
// Validate e-mail
```

```
if (!filter_var($email, FILTER_VALIDATE_EMAIL) === false) {
```

```
    echo("$email is a valid email address");
```

```
} else {
```

```
    echo("$email is not a valid email address");
```

```
}
```

```
?>
```



PHP Error Handling

- The default error handling in PHP is very simple. An error message with filename, line number and a message describing the error is sent to the browser.
- When creating scripts and web applications, error handling is an important part. If your code lacks error checking code, your program may look very unprofessional and you may be open to security risks.
- Different error handling methods:
 - Simple "die()" statements
 - Custom errors and error triggers
 - Error reporting



Basic Error Handling: Using the die() function

- The first example shows a simple script that opens a text file:

```
<?php
$file=fopen("welcome.txt","r");
?>
```

- If the file does not exist you might get an error like this:

Warning: *fopen(welcome.txt)*
[function.fopen]: failed to open
stream:No such file or directory
in C:\webfolder\test.php on line
2

- To prevent the user from getting an error message like the one above, we test whether the file exist before we try to access it:

```
<?php
if(!file_exists("welcome.txt")) {
    die("File not found");
} else {
    $file=fopen("welcome.txt","r");
}?>
```

- The code above is more efficient than the earlier code, because it uses a simple error handling mechanism to stop the script after the error



Creating a Custom Error Handler

- Creating a custom error handler is quite simple. We simply create a special function that can be called when an error occurs in PHP.
- This function must be able to handle a minimum of two parameters (error level and error message) but can accept up to five parameters (optionally: file, line-number, and the error context):
 - **error_level**-Required. Specifies the error report level for the user-defined error. Must be a value number.
 - **error_message**-Required. Specifies the error message for the user-defined error.
 - **error_file**- Optional. Specifies the filename in which the error occurred
 - **error_line** -Optional. Specifies the line number in which the error occurred
 - **error_context**-Optional. Specifies an array containing every variable, and their values, in use when the error occurred

Syntax

```
error_function(error_level,error_message,  
error_file,error_line,error_context)
```




Error Report levels

These error report levels are the different types of error the user-defined error handler can be used for:

- **2 – E_WARNING**- on-fatal run-time errors. Execution of the script is not halted
- **8 - E_NOTICE**- Run-time notices. The script found something that might be an error, but could also happen when running a script normally
- **256- E_USER_ERROR**- Fatal user-generated error. This is like an E_ERROR set by the programmer using the PHP function `trigger_error()`
- **512- E_USER_WARNING**- Non-fatal user-generated warning. This is like an E_WARNING set by the programmer using the PHP function `trigger_error()`
- **1024-E_USER_NOTICE**-User-generated notice. This is like an E_NOTICE set by the programmer using the PHP function `trigger_error()`
- **4096-E_RECOVERABLE_ERROR**-Catchable fatal error. This is like an E_ERROR but can be caught by a user defined handle (see also `set_error_handler()`)
- **8191-E_ALL**-All errors and warnings (E_STRICT became a part of E_ALL in PHP 5.4)



Now lets create a function to handle errors:

```
function customError($errno, $errstr) {  
    echo "<b>Error:</b> [$errno] $errstr<br>";  
    echo "Ending Script";  
    die();  
}
```

- The code above is a simple error handling function. When it is triggered, it gets the error level and an error message. It then outputs the error level and message and terminates the script.
- Now that we have created an error handling function we need to decide when it should be triggered.



Set Error Handler

- The default error handler for PHP is the built in error handler. We are going to make the function above the default error handler for the duration of the script.
- It is possible to change the error handler to apply for only some errors, that way the script can handle different errors in different ways. However, in this example we are going to use our custom error handler for all errors:

```
set_error_handler("customError");
```

Example

- Testing the error handler by trying to output variable that does not exist:

```
<?php  
    //error handler function  
    function customError($errno,  
        $errstr) {  
        echo "<b>Error:</b> [$errno]  
        $errstr";  
    }  
    //set error handler  
    set_error_handler("customError");  
    //trigger error  
    echo($test);  
?>
```



