COMP30680
Web Application Development

PHP part 3 – file management.

David Coyle
d.coyle@ucd.ie

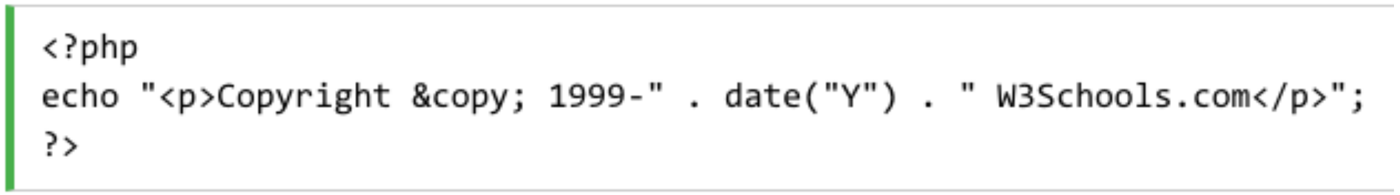
PHP Include / Require

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.

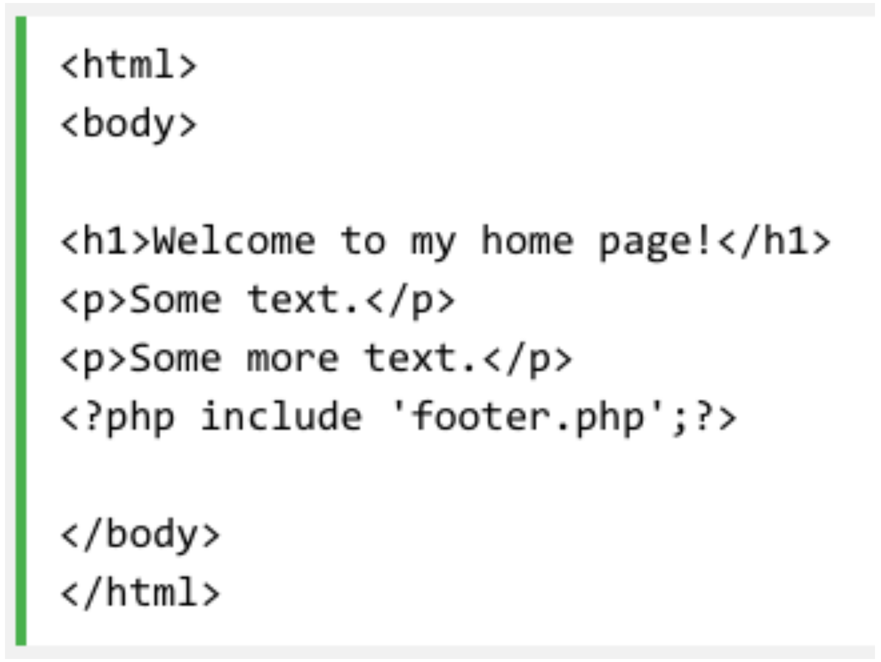
It is possible to insert the content of one PHP file into another PHP file (before the server executes it), with the include or require statement.

Including files is very useful when you want to include the same PHP, HTML, or text on multiple pages of a website.

In the example here a PHP file containing footer information can be included easily in other PHP files.



```
<?php
echo "<p>Copyright &copy; 1999-" . date("Y") . " W3Schools.com</p>";
?>
```



```
<html>
<body>

<h1>Welcome to my home page!</h1>
<p>Some text.</p>
<p>Some more text.</p>
<?php include 'footer.php';?>

</body>
</html>
```

See **include_footer.php**

PHP Include / Require

Include can also used to add common features like menus:

```
<?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>';
?>
```

```
<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>
```

See **include_menu.php**

PHP Include or Require

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.

If you want the execution to go on and show users the output, even if the include file is missing, use the include statement.

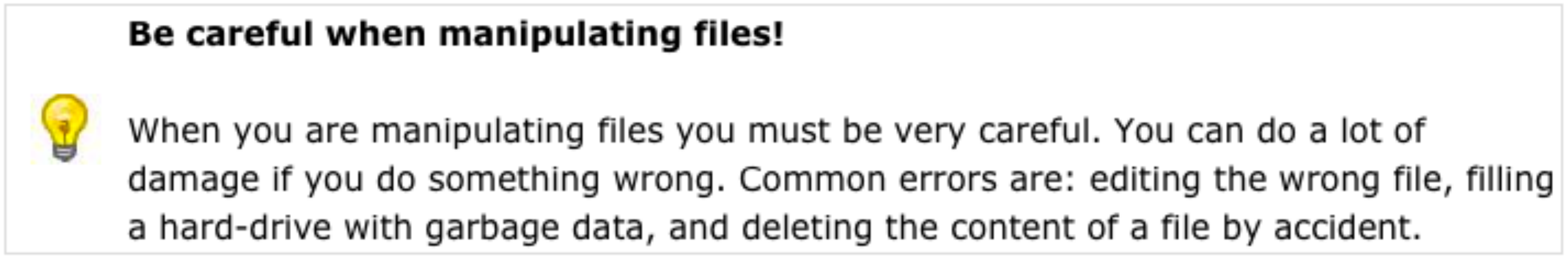
Otherwise it is best to use the require statement to include a key file to the flow of execution.

This is particularly important is your external files have complex PHP application coding, e.g. security features. Using require will help avoid compromising your application's security and integrity, just in-case one key file is accidentally missing.

PHP Manipulating Files

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

PHP has several functions for creating, reading, uploading, and editing files.



Be careful when manipulating files!

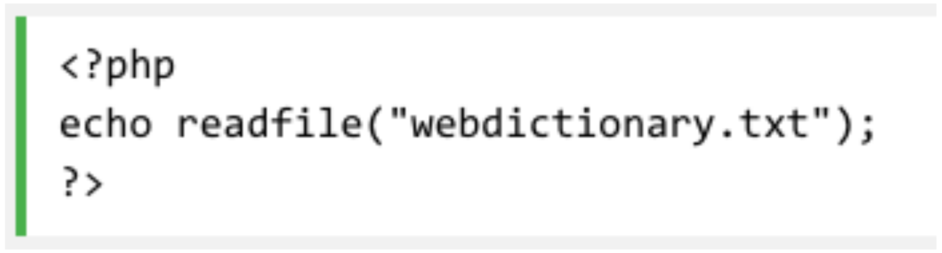
When you are manipulating files you must be very careful. You can do a lot of damage if you do something wrong. Common errors are: editing the wrong file, filling a hard-drive with garbage data, and deleting the content of a file by accident.

w3schools.com

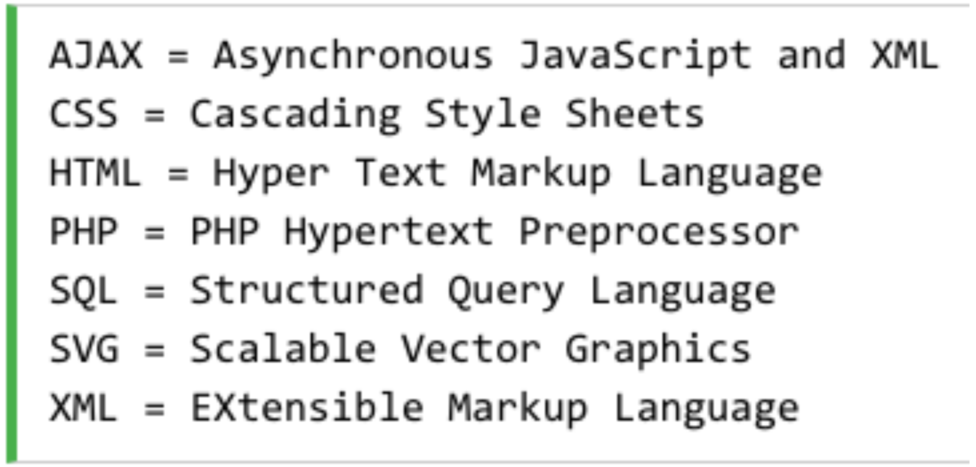
Basic file read

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

For example the following code will open a text file called "webdictionary.txt", stored on the server, and read its contents:



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



```
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
```

See **basic_readFile.php**

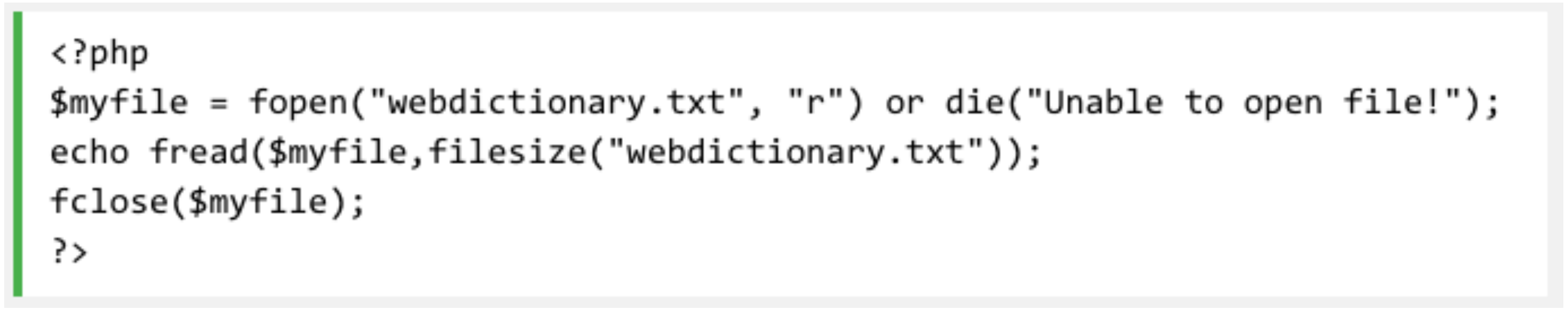
Note: readfile() is ok if all you want to do is open up a file and read its contents.

Open, Read and Close a file

A better method to open files is with the **fopen()** function, read it using **fread()** and then close it with **fclose()**.

This approach gives you more options than the readfile() function.

Start with **fopen()**:



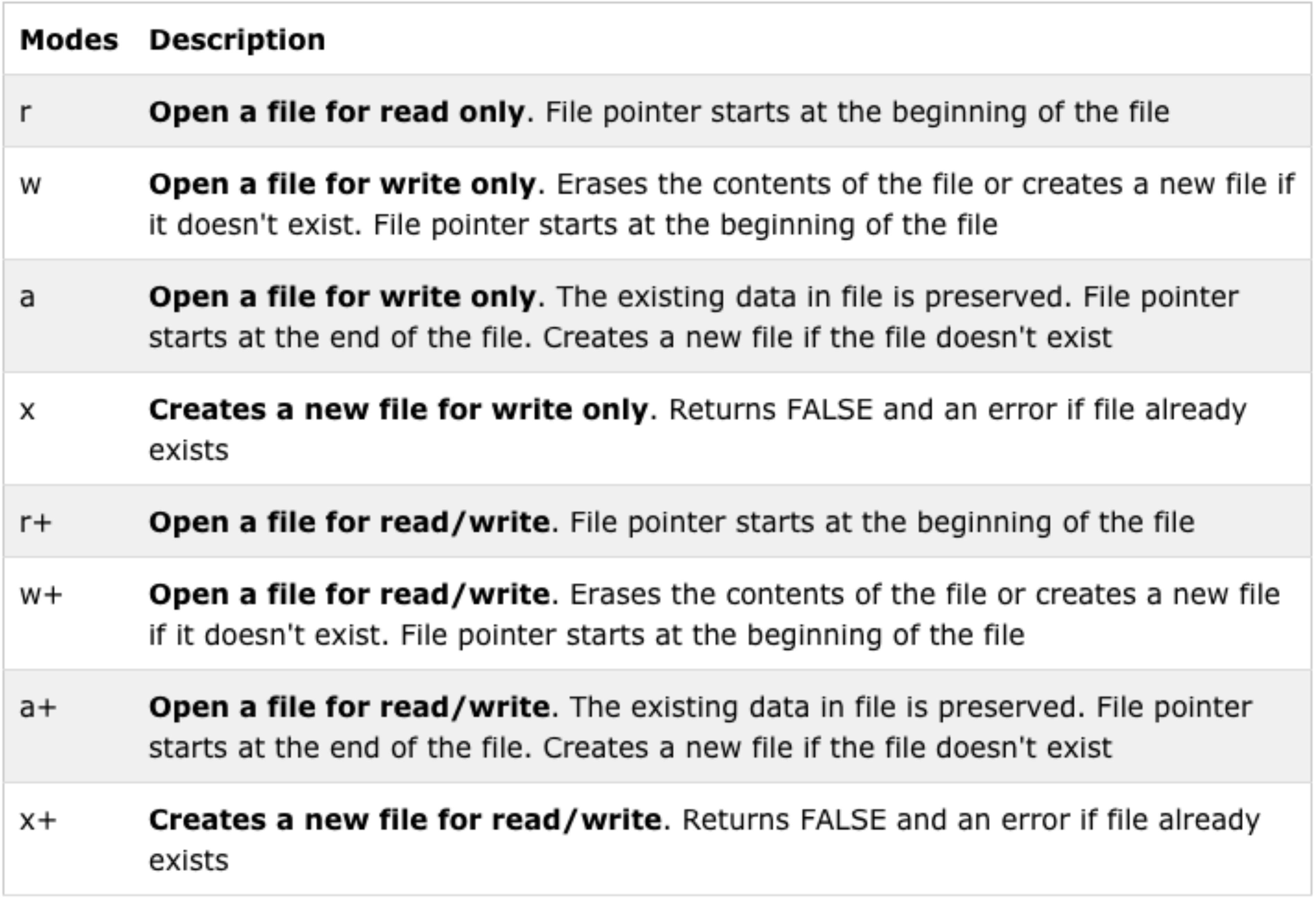
```
<?php
$myfile = fopen("webdictionary.txt", "r") or die("Unable to open file!");
echo fread($myfile,filesize("webdictionary.txt"));
fclose($myfile);
?>
```

The first parameter of fopen() contains the name of the file to be opened.

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.

fopen() modes



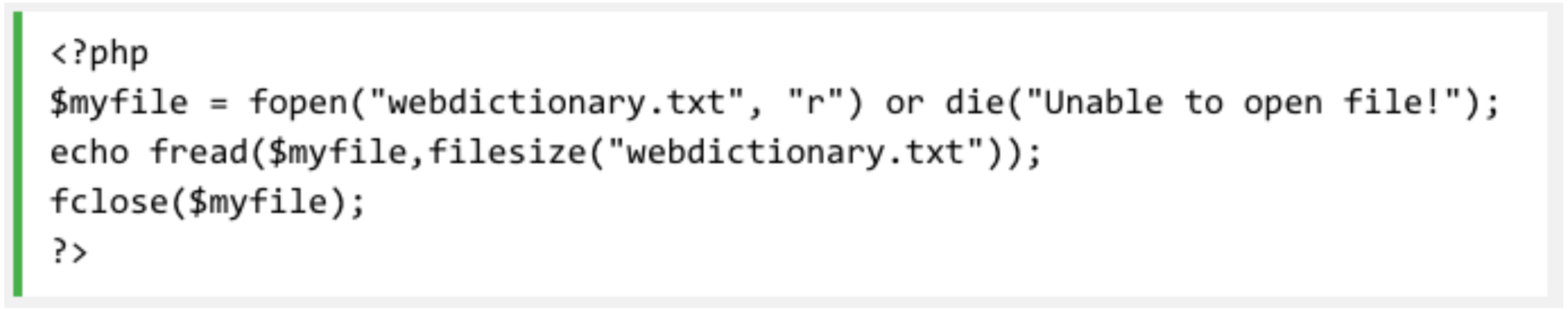
Modes	Description
r	**Open a file for read only**. File pointer starts at the beginning of the file
w	**Open a file for write only**. Erases the contents of the file or creates a new file if it doesn't exist. File pointer starts at the beginning of the file
a	**Open a file for write only**. The existing 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

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.

In our example the PHP code reads the "webdictionary.txt" file to the end.

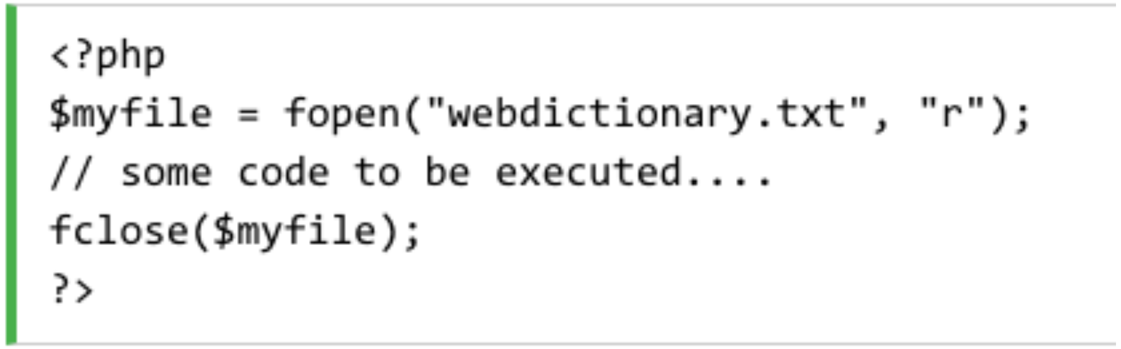


```php
<?php
$myfile = fopen("webdictionary.txt", "r") or die("Unable to open file!");
echo fread($myfile,filesize("webdictionary.txt"));
fclose($myfile);
?>
```

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) you want to close.



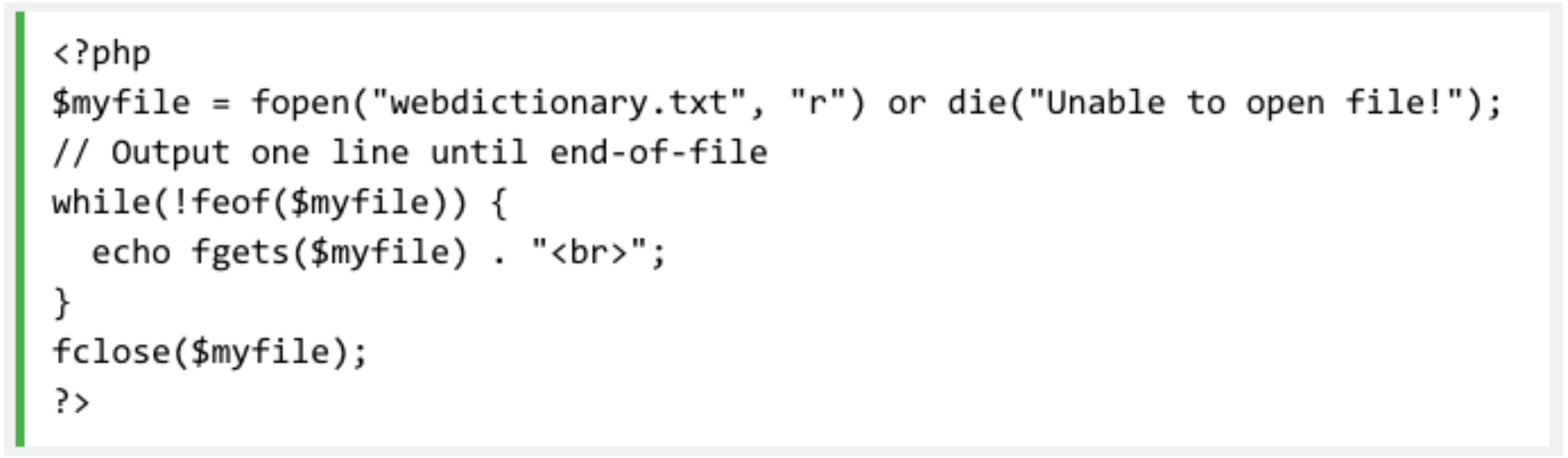
```
<?php
$myfile = fopen("webdictionary.txt", "r");
// some code to be executed....
fclose($myfile);
?>
```

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

Further examples of file manipulation

fgets() - used to read a single line from a file. After a call to the fgets() function, the file pointer has moves to the next line.

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



```
<?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);
?>
```

See **read_lineByLine.php**

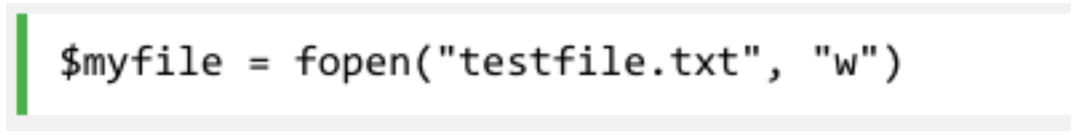
For a complete reference of PHP filesystem functions see: http://www.w3schools.com/php/php_ref_filesystem.asp

Create a file

The **fopen()** function is also used to create a file.

If you use fopen() on a file that does not exist, it will create it.

The following create a new file called "testfile.txt". The file is created in the same directory where the PHP code resides:



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

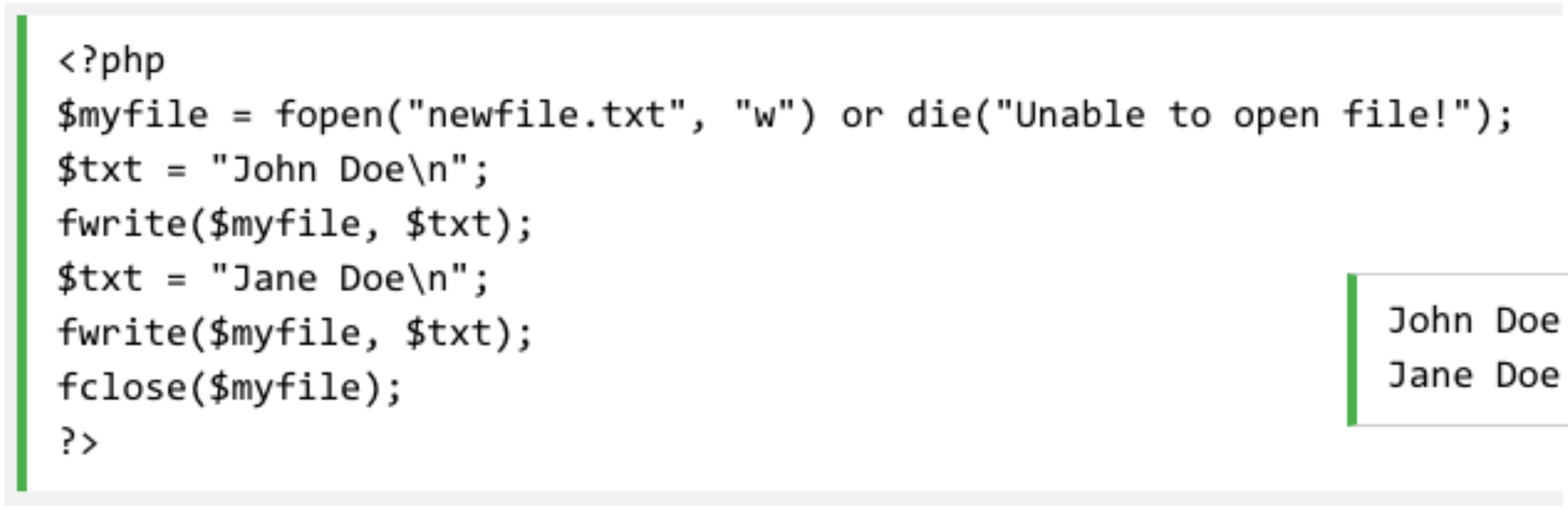
Note the mode: (w). You can also use append (a).

Write to a file

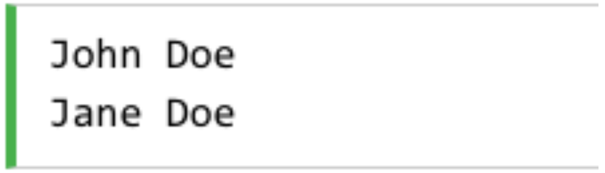
The **fwrite()** function is used to write to a file.

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

This example writes some of names into a new file called "newfile.txt":



```php
<?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);
?>
```



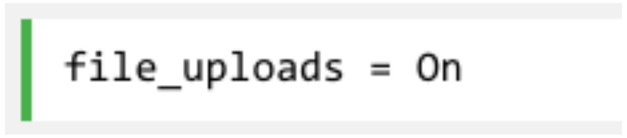
```
John Doe
Jane Doe
```

Note: If you opened newfile.txt again and wrote new text/data to it all the existing data will be ERASED.

Uploading a file

First check the "php.ini" file.

In your "php.ini" file, search for the file_uploads directive, and set it to On.



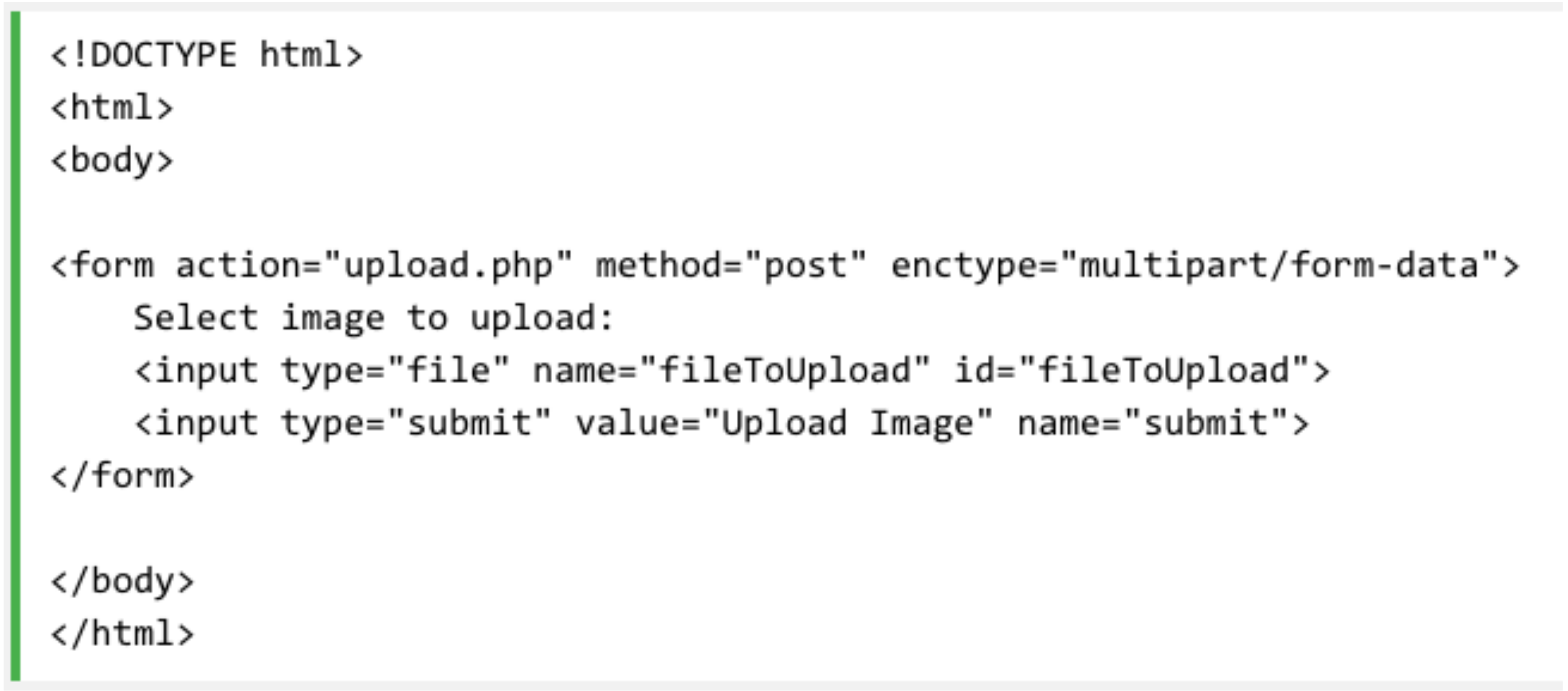
```
file_uploads = On
```

With PHP, it is easy to upload files to the server.

However, with ease comes danger, so always be careful when allowing file uploads!

Next we walk through a step by step example, uploading an image file.

Uploading step 1 – create a HTML form.



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

<form action="upload.php" method="post" enctype="multipart/form-data">
    Select image to upload:
    <input type="file" name="fileToUpload" id="fileToUpload">
    <input type="submit" value="Upload Image" name="submit">
</form>

</body>
</html>
```

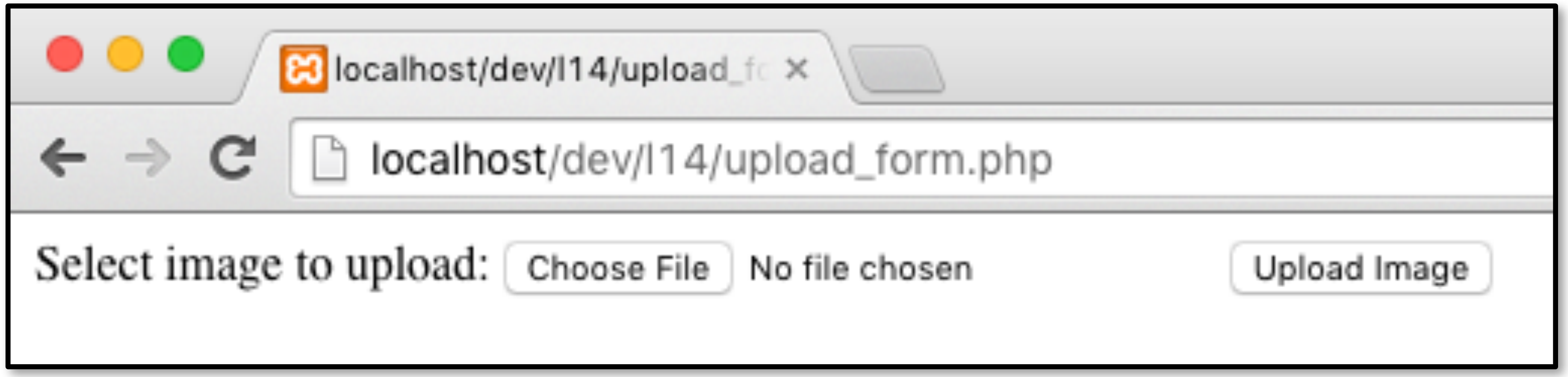
Some rules to follow for the HTML form above:

- Make sure that the form uses method="post".
- The form also needs the following attribute: enctype="multipart/form-data". It specifies which content-type to use when submitting the form.

Without these requirements, the file upload will not work.

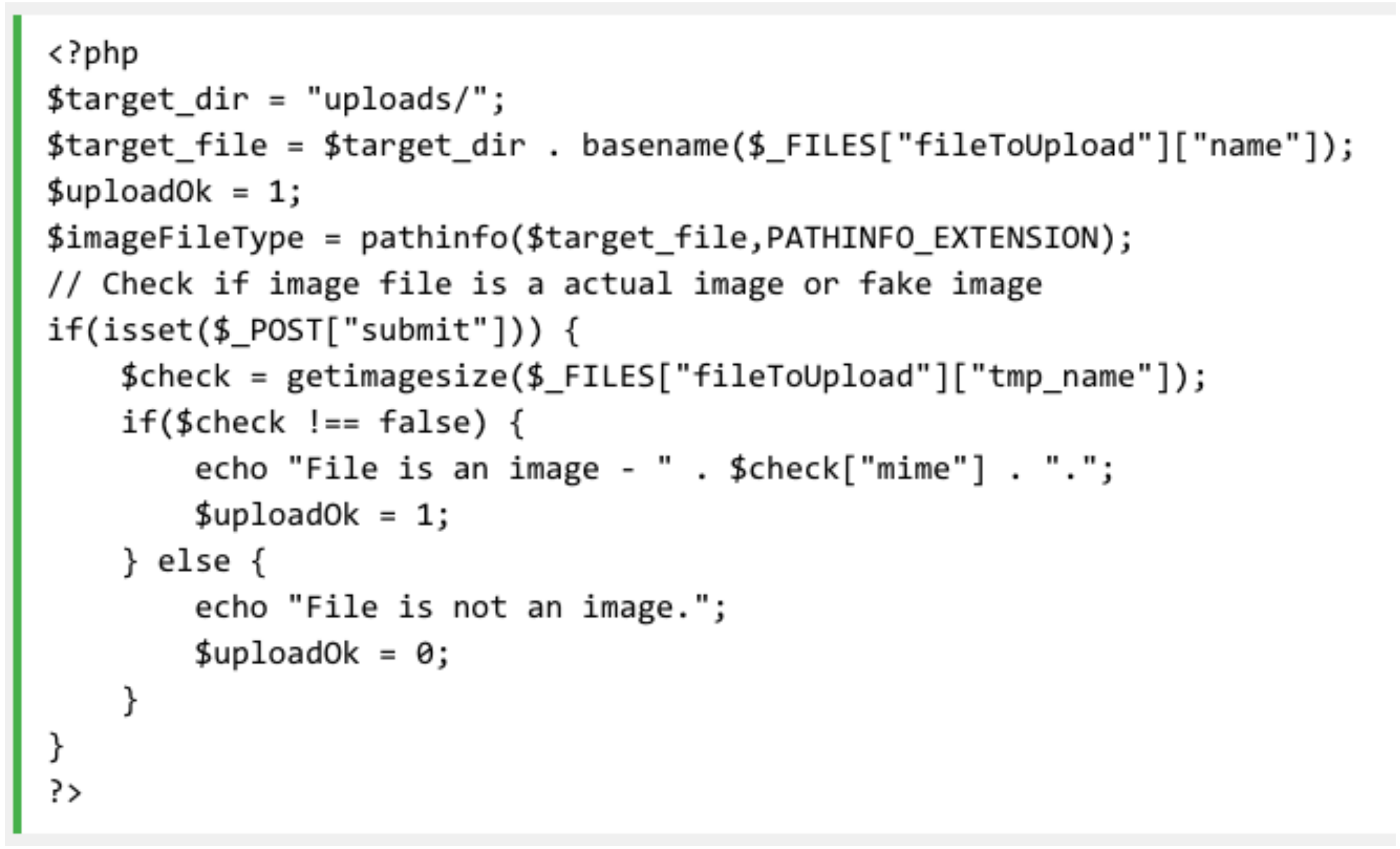
Uploading step 1 – create a HTML form.

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

<form action="upload.php" method="post" enctype="multipart/form-data">
    Select image to upload:
    <input type="file" name="fileToUpload" id="fileToUpload">
    <input type="submit" value="Upload Image" name="submit">
</form>

</body>
</html>
```


localhost/dev/l14/upload_fo
localhost/dev/l14/upload_form.php
Choose File
Upload Image

Select image to upload: No file chosen

Uploading step 2 – upload.php



```php
<?php
$target_dir = "uploads/";
$target_file = $target_dir . basename($_FILES["fileToUpload"]["name"]);
$uploadOk = 1;
$imageFileType = pathinfo($target_file,PATHINFO_EXTENSION);
// Check if image file is a actual image or fake image
if(isset($_POST["submit"])) {
    $check = getimagesize($_FILES["fileToUpload"]["tmp_name"]);
    if($check !== false) {
        echo "File is an image - " . $check["mime"] . ".";
        $uploadOk = 1;
    } else {
        echo "File is not an image.";
        $uploadOk = 0;
    }
}
?>
```

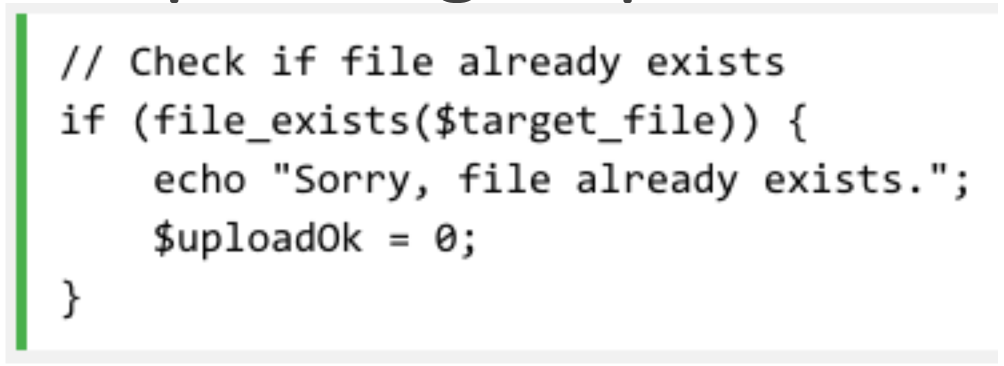
Uploading step 2 – upload.php

The "upload.php" file contains the code for uploading a file.

- $target_dir = "uploads/" - specifies the directory where the file is going to be placed **
- $target_file specifies the path of the file to be uploaded
- $uploadOk=1 is not used yet (will be used later)
- $imageFileType holds the file extension of the file
- Next, check if the image file is an actual image or a fake image

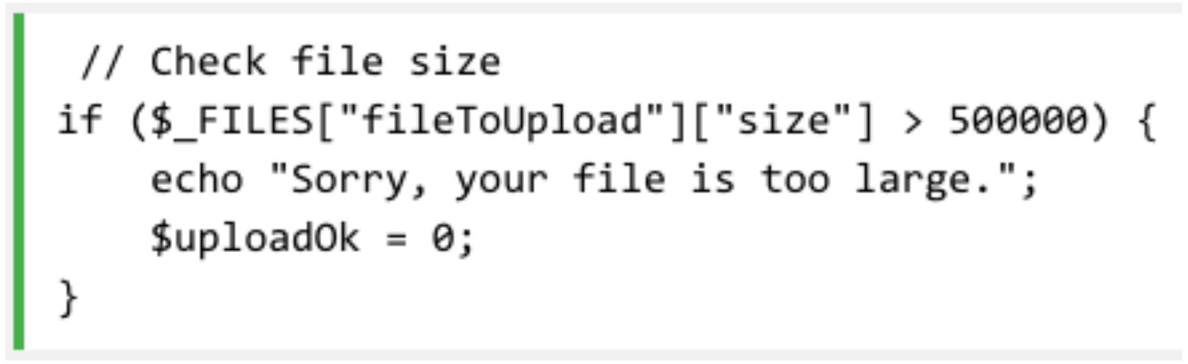
**** Note:** You will need to create a new directory called "uploads" in the directory where "upload.php" file resides. The uploaded files will be saved there.

Uploading step 3 – additional checks



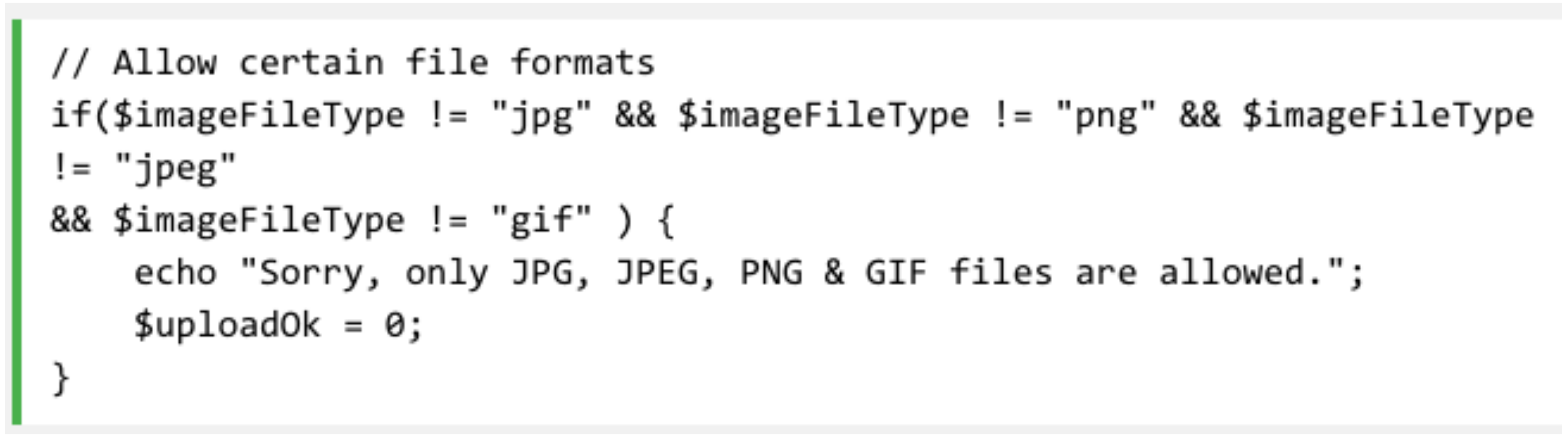
```
// Check if file already exists
if (file_exists($target_file)) {
    echo "Sorry, file already exists.";
    $uploadOk = 0;
}
```

Check if the file already exists in the "uploads" folder. If it does, an error message is displayed, and $uploadOk is set to 0:



```
 // Check file size
if ($_FILES["fileToUpload"]["size"] > 500000) {
    echo "Sorry, your file is too large.";
    $uploadOk = 0;
}
```

Check the size of the file. If the file is larger than 500kb, an error message is displayed, and $uploadOk is set to 0



```
// Allow certain file formats
if($imageFileType != "jpg" && $imageFileType != "png" && $imageFileType
!= "jpeg"
&& $imageFileType != "gif" ) {
    echo "Sorry, only JPG, JPEG, PNG & GIF files are allowed.";
    $uploadOk = 0;
}
```

Only allow users to upload JPG, JPEG, PNG, and GIF files. All other file types gives an error message before setting $uploadOk to 0:

Uploading complete.

See **upload.php**

Test this file and modify it for different file types etc.

For a complete reference of filesystem functions, go to our complete PHP Filesystem Reference.

Questions, Suggestions?

Materials and further reading:

w3Schools advanced PHP tutorials: http://www.w3schools.com/php/php_includes.asp

Next:

Error handling