

PHP ADVANCED PROGRAMMING I

WEN-YEN WANG

ABOUT THE SKILLS YOU MUST USE IN PROGRAMMING

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.
- 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.

Syntax


```
include 'filename';  
  
or  
  
require 'filename';
```

EXAMPLE 1 FOR INCLUDE STATEMENT

- **ex61-02.php** invokes **ex61-01.php**.

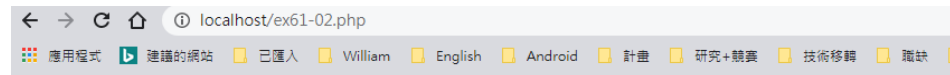
```
ex61-02.php
<!DOCTYPE html>
<html>
<body>
<h1>Welcome to my home page!</h1>
<p>Some text.</p>
<p>Some more text.</p>
<?php include 'ex61-01.php';?>
</body>
</html>
```

```
ex61-01.php
1 <?php
2 echo "<p>Copyright &copy; 1999-" . date("Y") . " W3Schools.com</p>";
3 ?>
```



EXAMPLE 1 FOR INCLUDE STATEMENT

- **Correct one!**



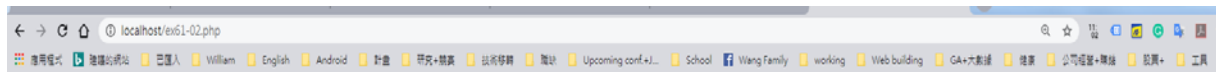
Welcome to my home page!

Some text.

Some more text.

Copyright © 1999-2020 W3Schools.com

- **Wrong one with warning messages!**



Welcome to my home page!

Some text.

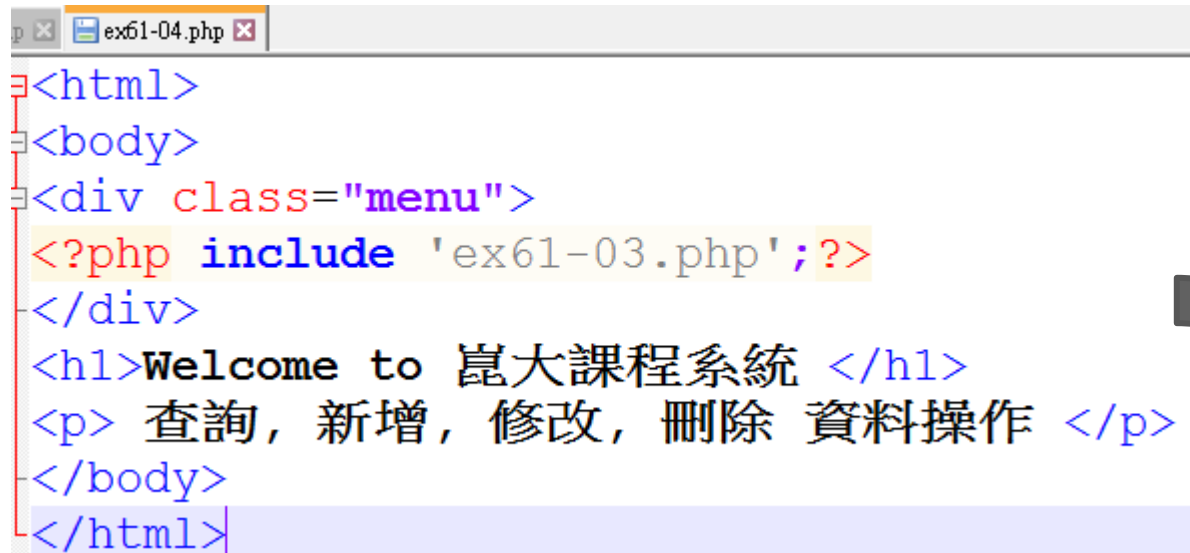
Some more text.

Warning: include(ex61-1.php): failed to open stream: No such file or directory in C:\xampp\htdocs\ex61-02.php on line 7

Warning: include(): Failed opening 'ex61-1.php' for inclusion (include_path='C:\xampp\php\PEAR') in C:\xampp\htdocs\ex61-02.php on line 7

EXAMPLE 2 FOR INCLUDE STATEMENT

- **ex61-04.php** invokes **ex61-03.php**.



A screenshot of a code editor window titled 'ex61-04.php'. The code is written in PHP and HTML. It includes an include statement for 'ex61-03.php'. The code is as follows:

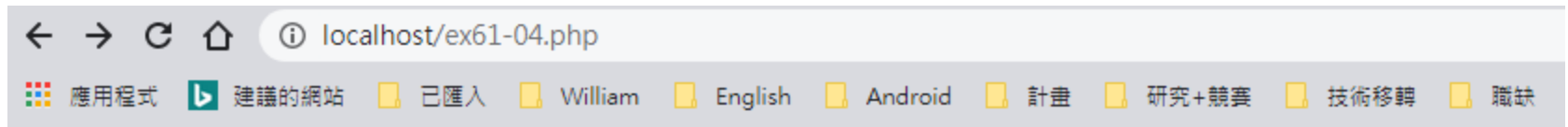
```
<html>
<body>
<div class="menu">
  <?php include 'ex61-03.php';?>
</div>
<h1>Welcome to 崑大課程系統 </h1>
<p> 查詢, 新增, 修改, 刪除 資料操作 </p>
</body>
</html>
```



A screenshot of a code editor window titled 'ex61-03.php'. The code is written in PHP and HTML. It contains a series of echo statements that output navigation links. The code is as follows:

```
1 <?php
2 echo '<a href="/default.php">Home</a> -
3 <a href="/html/default.php"> 查詢 </a> -
4 <a href="/css/default.php"> 新增 </a> -
5 <a href="/js/default.php"> 修改 </a> -
6 <a href="default.php"> 刪除 </a>';
7 ?>
```

EXAMPLE 2 FOR INCLUDE STATEMENT



[Home](#) - [查詢](#) - [新增](#) - [修改](#) - [刪除](#)

Welcome to 崑大課程系統

查詢, 新增, 修改, 刪除 資料操作

EXAMPLE 3 FOR INCLUDE STATEMENT

- **The variables in called files can be used in the calling file.**

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

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

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.

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

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

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

</body>
</html>
```

Welcome to my home page!

I have a red BMW.

PHP FILE OPEN/READ/CLOSE

- 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.

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

ex61-05.php webdictionary.txt

```
1 <?php
2 $myfile = fopen("webdictionary.txt", "r")
3           or die("Unable to open file!");
4 echo fread($myfile,filesize("webdictionary.txt"));
5 fclose($myfile);
6 ?>
```

Output

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

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.

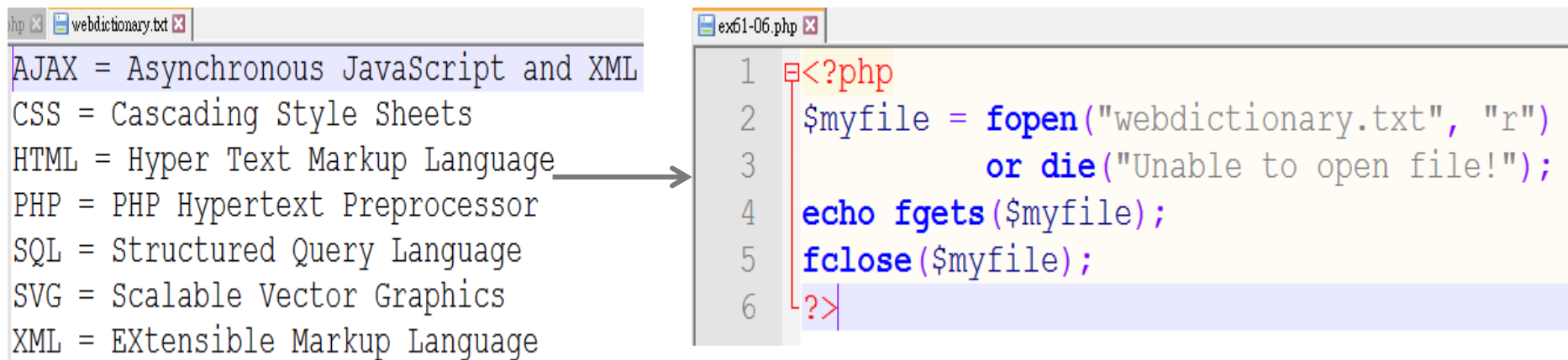
It's a good programming practice to close all files after you have finished with them. You don't want an open file running around on your server taking up resources!

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()

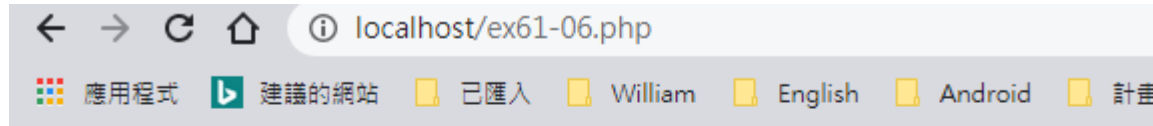
The `fgets()` function is used to **read a single line** from a file. The example below outputs the first line of the "webdictionary.txt" file:



```
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

ex61-06.php
1 <?php
2 $myfile = fopen("webdictionary.txt", "r")
3     or die("Unable to open file!");
4 echo fgets($myfile);
5 fclose($myfile);
6 ?>
```

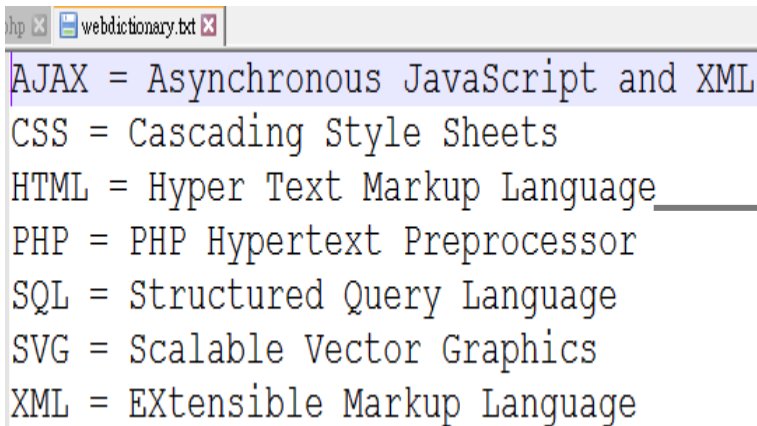
Output



AJAX = Asynchronous JavaScript and XML

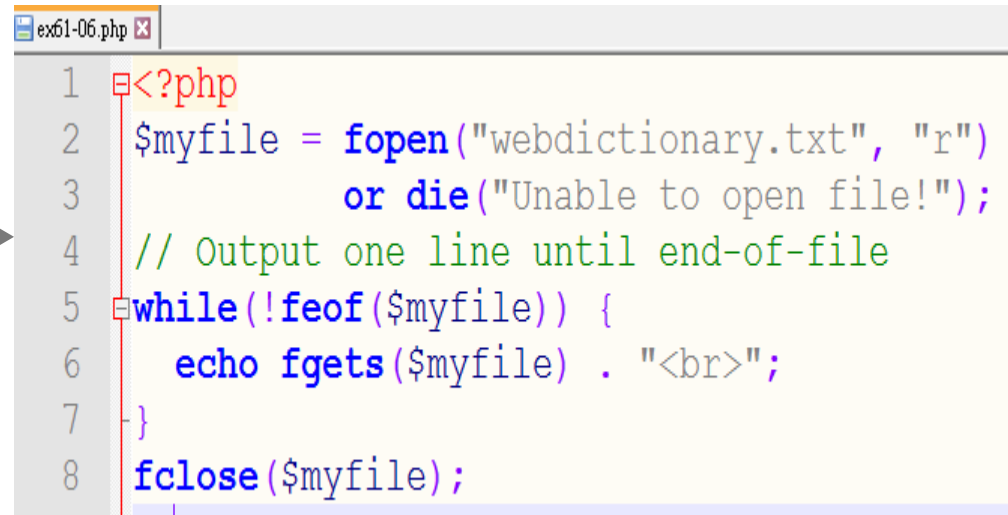
PHP CHECK END-OF-FILE - FEOF()+ FGETS()

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:



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

Output



```
1 <?php
2 $myfile = fopen("webdictionary.txt", "r")
3         or die("Unable to open file!");
4 // Output one line until end-of-file
5 while(!feof($myfile)) {
6     echo fgetc($myfile) . "<br>";
7 }
8 fclose($myfile);
```

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

PHP READ SINGLE CHARACTER - FEOF()+ 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);
?>
```

Output

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

PHP READ SINGLE CHARACTER - FEOF()+ FGETC()

```
<?php
$myfile = fopen("webdictionary.txt", "r")
          or die("Unable to open file!");
// Output one line until end-of-file
$i = 1;
while(!feof($myfile)) {
    //echo "$i: " ;
    echo fgetc($myfile) ;
    //echo "<br>";
    $i++;
}
fclose($myfile);
?>
```

PHP FOPEN-FWRITE

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 FOPEN-FWRITE

PHP File Permissions

If you are having errors when trying to get this code to run, check that you have granted your PHP file access to write information to the hard drive.

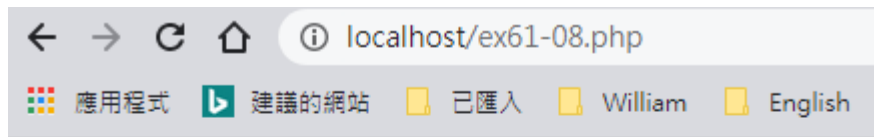
PHP Write to File - fwrite()

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

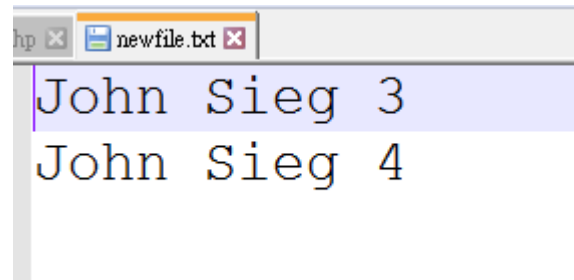
The first parameter of `fwrite()` contains the name of the file to write to and the second parameter is the string to be written.

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

PHP FOPEN-FWRITE



The execution is done completely!



PHP FOPEN-FWRITE

```
ex61-08.php x newfile.txt x
1  <?php
2
3      $myfile = fopen("newfile.txt", "w")
4          or die("Unable to open file!");
5      $txt = "John Sieg 3\n";
6      fwrite($myfile, $txt)
7          or die("Unable to write to file!");
8      $txt = "John Sieg 4\n";
9      fwrite($myfile, $txt)
10         or die("Unable to write to file!");
11      fclose($myfile);
12      echo "The execution is done completely! <br>"
13  ?>
```

P.S. be careful! fopen(), fwrite() and fclose() work together, and clear out everything in the output file, then write something into the file.

MORE TALKS ABOUT FILES AND SOMETHING

PHP FUNCTIONS

PHP FUNCTIONS

The real power of PHP comes from its functions.

PHP has more than 1000 built-in functions, and in addition you can create your own custom functions.

PHP Built-in Functions

PHP has over 1000 built-in functions that can be called directly, from within a script, to perform a specific task.

Please check out our PHP reference for a complete overview of the [PHP built-in functions](#).

PHP Reference

The PHP reference contains different categories of all PHP functions and constants, along with examples.

Array	Calendar	Date	Directory	Error
Filesystem	Filter	FTP	Libxml	Mail
Math	Misc	MySQLi	Network	SimpleXML
Stream	String	XML Parser	Zip	Timezones

PHP USER DEFINED FUNCTIONS

Besides the built-in PHP functions, it is possible to create your own functions.

- A function is a block of statements that can be used repeatedly in a program.
- A function will not execute automatically when a page loads.
- A function will be executed by a call to the function.

Create a User Defined Function in PHP

A user-defined function declaration starts with the word `function`:

Syntax

```
function functionName() {  
    code to be executed;  
}
```

Note: A function name must start with a letter or an underscore. Function names are NOT case-sensitive.

PHP USER DEFINED FUNCTIONS

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

<?php
function writeMsg() {
    echo "Hello world!";
}

writeMsg();
?>

</body>
</html>
```

Hello world!

PHP USER DEFINED ARGUMENTS

Information can be passed to functions through arguments. An argument is just like a variable.

Arguments are specified after the function name, inside the parentheses. You can add as many arguments as you want, just separate them with a comma.

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

<?php
function familyName($fname) {
    echo "$fname Refsnes.<br>";
}

familyName("Jani");
familyName("Hege");
familyName("Stale");
familyName("Kai Jim");
familyName("Borge");
?>

</body>
</html>
```

```
Jani Refsnes.
Hege Refsnes.
Stale Refsnes.
Kai Jim Refsnes.
Borge Refsnes.
```

PHP USER DEFINED ARGUMENTS

The following example has a function with two arguments (\$fname and \$year):

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

<?php
function familyName($fname, $year) {
    echo "$fname Refsnes. Born in $year <br>";
}

familyName("Hege", "1975");
familyName("Stale", "1978");
familyName("Kai Jim", "1983");
?>

</body>
</html>
```

Hege Refsnes. Born in 1975
Stale Refsnes. Born in 1978
Kai Jim Refsnes. Born in 1983

PHP IS A LOOSELY TYPED LANGUAGE

In the example above, notice that we did not have to tell PHP which data type the variable is.

PHP automatically associates a data type to the variable, depending on its value. Since the data types are not set in a strict sense, you can do things like adding a string to an integer without causing an error.

In PHP 7, type declarations were added. This gives us an option to specify the expected data type when declaring a function, and by adding the `strict` declaration, it will throw a "Fatal Error" if the data type mismatches.

In the following example we try to send both a number and a string to the function without using `strict`:

Example

```
<?php
function addNumbers(int $a, int $b) {
    return $a + $b;
}
echo addNumbers(5, "5 days");
// since strict is NOT enabled "5 days" is changed to int(5), and it will return 10
?>
```

應用程式 建議的網站 已匯入 William English Android 計畫 研究+競賽 技術移轉 職缺 Upcoming conf.+J... School Wang Family

Notice: A non well formed numeric value encountered in C:\xampp\htdocs\QQ.php on line 2

PHP WITH DECLARATION

To specify `strict` we need to set `declare(strict_types=1);`. This must be on the very first line of the PHP file.

Example

```
<?php declare(strict_types=1); // strict requirement

function addNumbers(int $a, int $b) {
    return $a + $b;
}
echo addNumbers(5, "5 days");
// since strict is enabled and "5 days" is not an integer, an error will be thrown
?>
```

Fatal error: Uncaught TypeError: Argument 2 passed to addNumbers() must be of the type int, string given, called in C:\xampp\htdocs\QQ.php on line 6 and defined in C:\xampp\htdocs\QQ.php:3 Stack trace: #0 C:\xampp\htdocs\QQ.php(6): addNumbers(5, '5 days') #1 {main} thrown in C:\xampp\htdocs\QQ.php on line 3

PHP DEFAULT ARGUMENT VALUE

The following example shows how to use a default parameter. If we call the function `setHeight()` without arguments it takes the default value as argument:

Example

```
<?php declare(strict_types=1); // strict requirement
function setHeight(int $minheight = 50) {
    echo "The height is : $minheight <br>";
}
```

```
setHeight(350);
setHeight(); // will use the default value of 50
setHeight(135);
setHeight(80);
?>
```



The height is : 350

The height is : 50

The height is : 135

The height is : 80

PHP FUNCTIONS - RETURNING VALUES

Example

```
<?php declare(strict_types=1); // strict requirement
function sum(int $x, int $y) {
    $z = $x + $y;
    return $z;
}

echo "5 + 10 = " . sum(5, 10) . "<br>";
echo "7 + 13 = " . sum(7, 13) . "<br>";
echo "2 + 4 = " . sum(2, 4);
?>
```



5 + 10 = 15

7 + 13 = 20

2 + 4 = 6

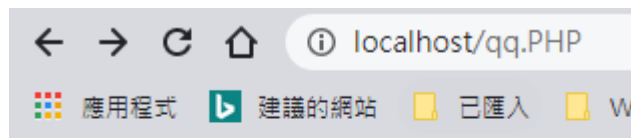
PHP RETURN TYPE DECLARATIONS

PHP 7 also supports Type Declarations for the `return` statement. Like with the type declaration for function arguments, by enabling the strict requirement, it will throw a "Fatal Error" on a type mismatch.

To declare a type for the function return, add a colon (`:`) and the type right before the opening curly (`{`) bracket when declaring the function.

Example

```
<?php declare(strict_types=1); // strict requirement
function addNumbers(float $a, float $b) : float {
    return $a + $b;
}
echo addNumbers(1.2, 5.2);
?>
```



PHP RETURN TYPE DECLARATIONS

Example

```
<?php declare(strict_types=1); // strict requirement
function addNumbers(float $a, float $b) : int {
    return (int)($a + $b);
}
echo addNumbers(1.2, 5.2);
?>
```

