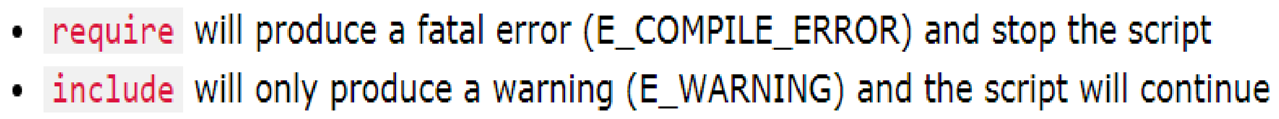
PHP ADVANCED PROGRAMMING I

WEN-YEN WANG

ABOUT THE SKILLS YOU MUST USE IN PROGRAMMING

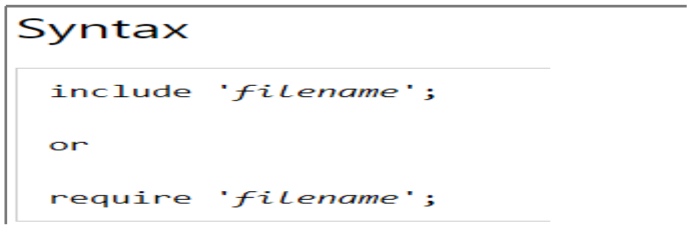
PHP INCLUDE FILES

- 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.

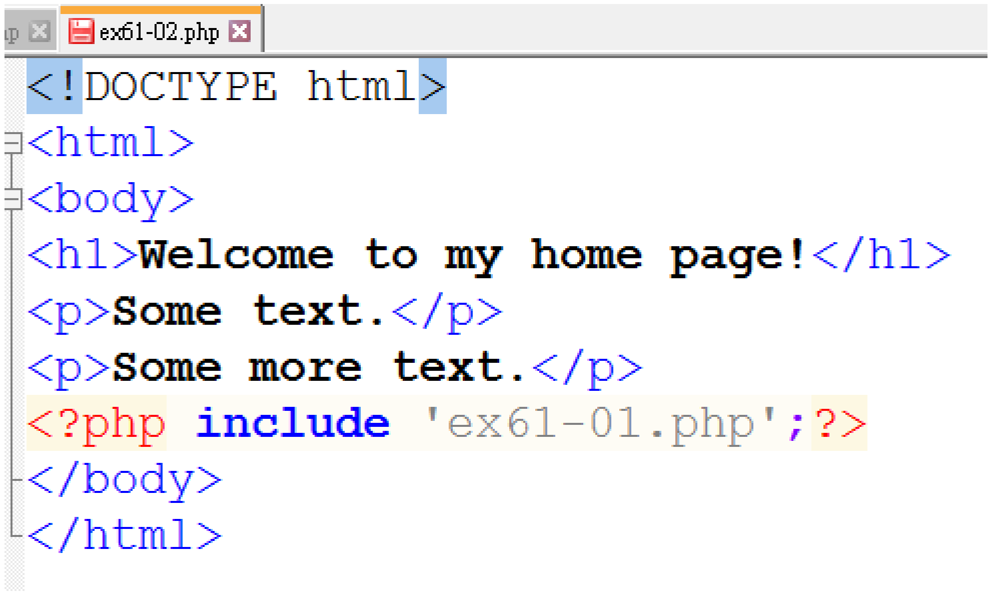


Syntax

```
include 'filename';

or

require 'filename';
```

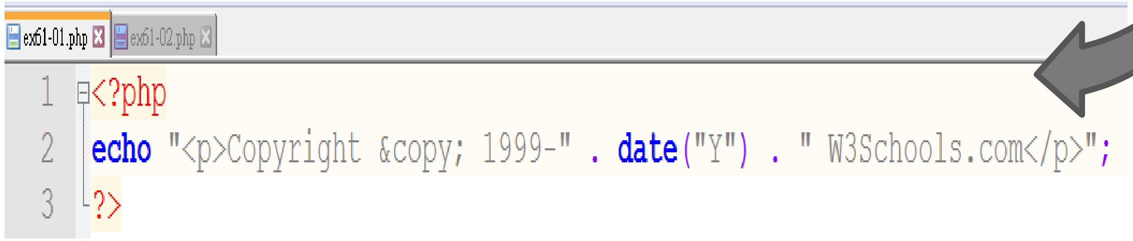
EXAMPLE 1 FOR INCLUDE STATEMENT

- **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>
```



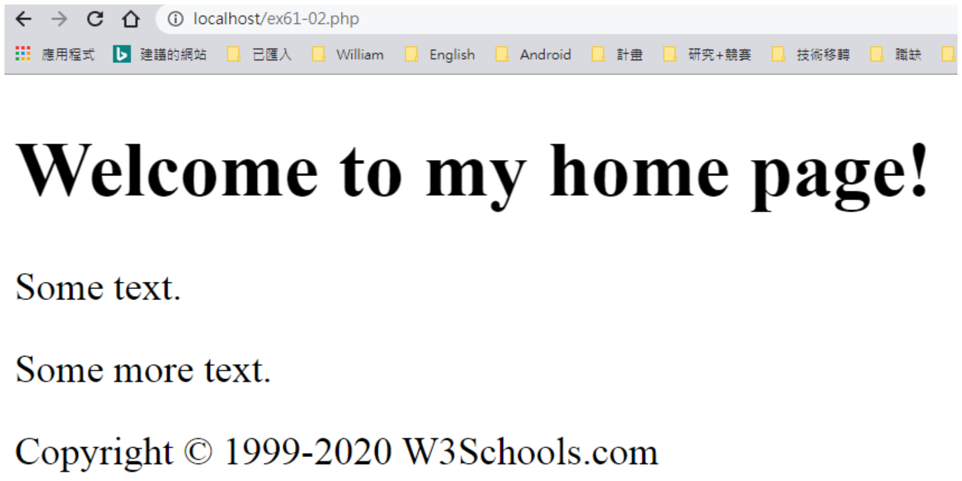
ex61-01.php ex61-02.php

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

ex61-01.php
ex61-02.php

EXAMPLE 1 FOR INCLUDE STATEMENT

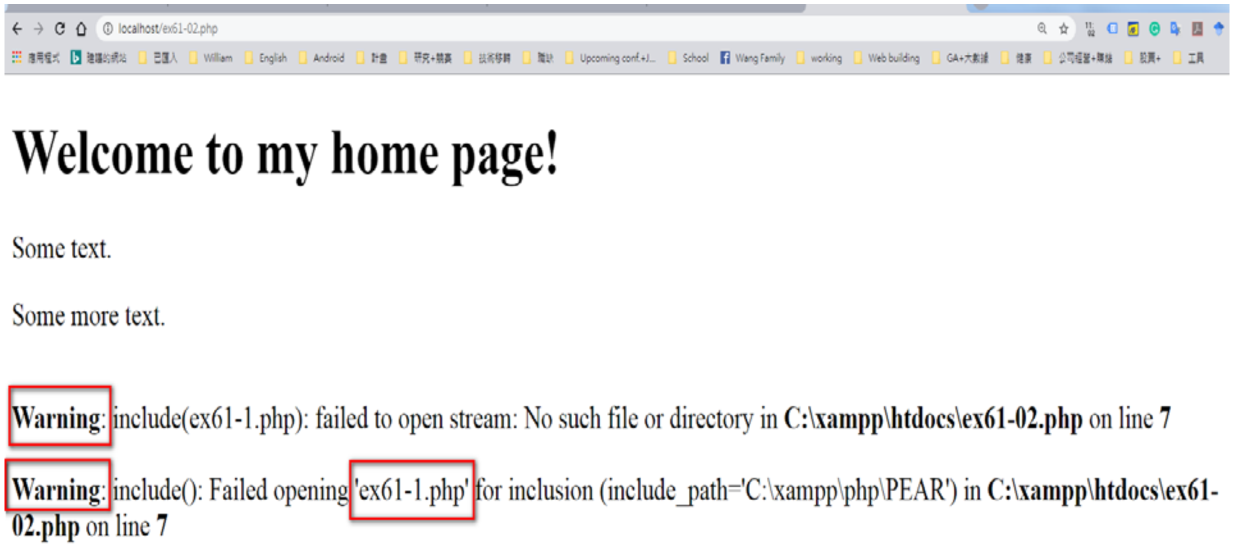
- **Correct one!**



localhost/ex61-02.php

```
Welcome to my home page!

Some text.

Some more text.

Copyright © 1999-2020 W3Schools.com
```

- **Wrong one with warning messages!**

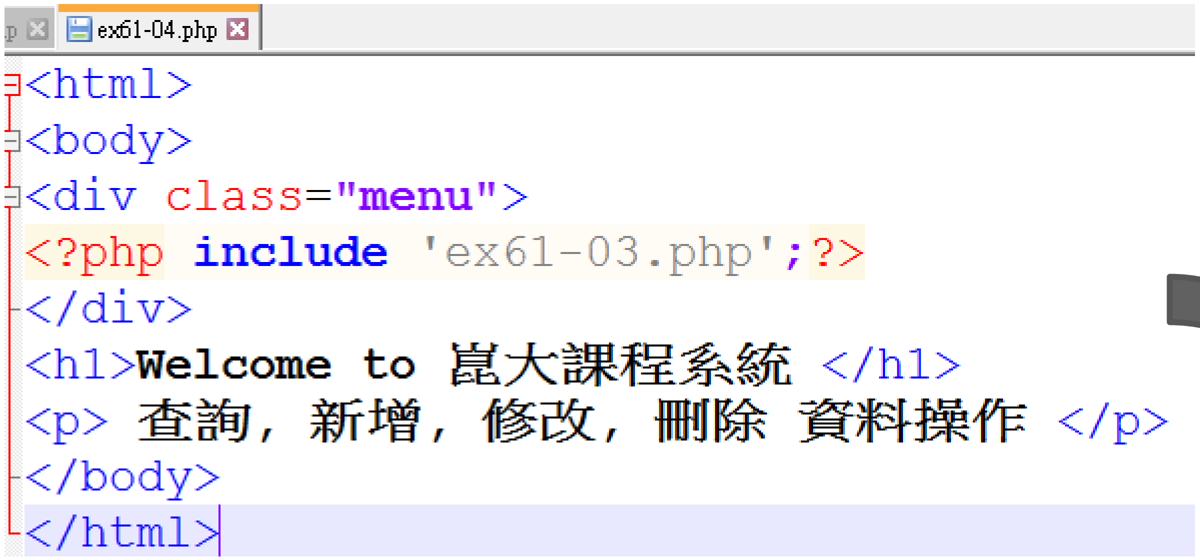


localhost/ex61-02.php

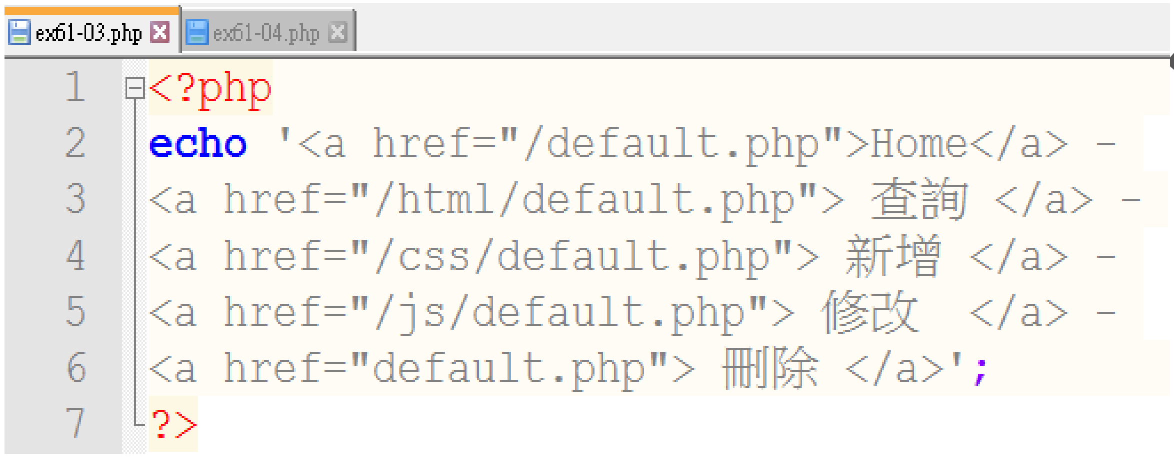
```
Welcome to my home page!

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.**



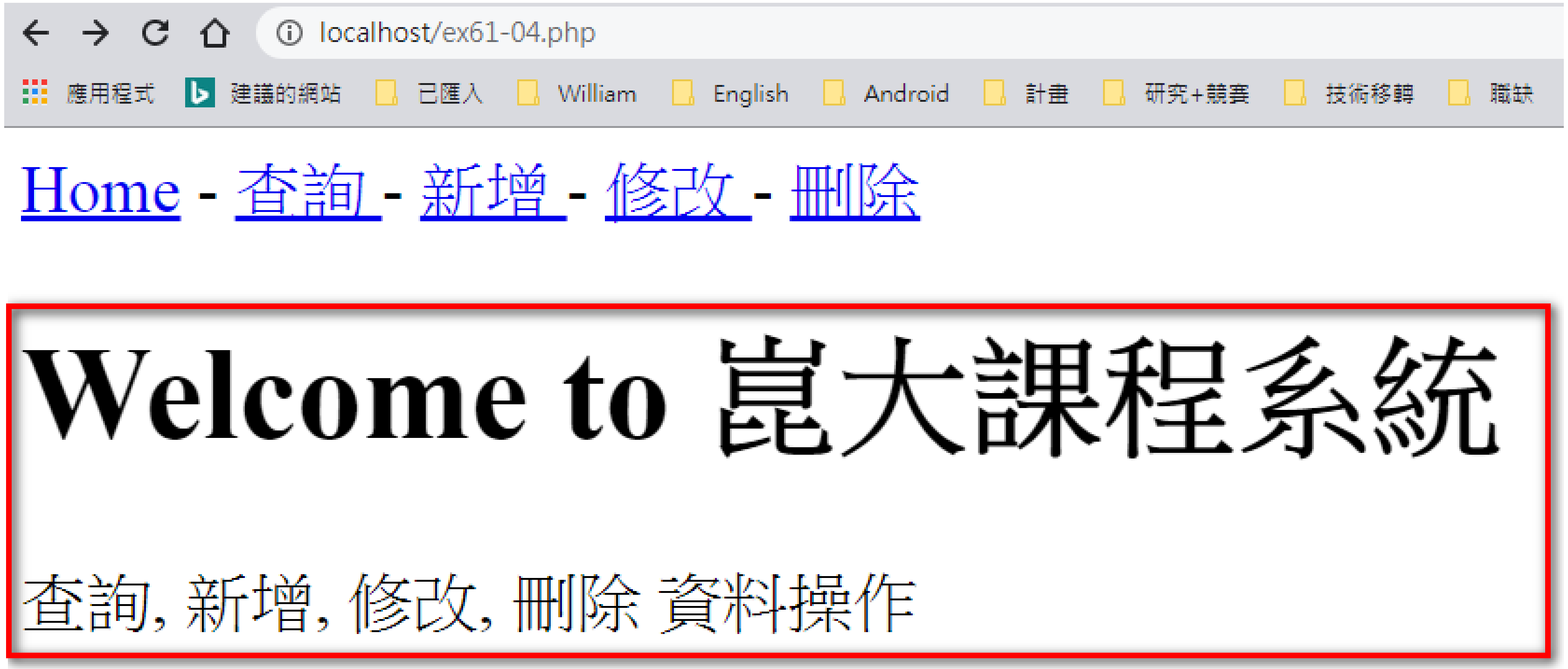
```
<html>
<body>
<div class="menu">
<?php include 'ex61-03.php';?>
</div>
<h1>Welcome to 崑大課程系統 </h1>
<p> 查詢， 新增， 修改， 刪除 資料操作 </p>
</body>
</html>
```



```
<?php
echo '<a href="/default.php">Home</a> -
<a href="/html/default.php"> 查詢 </a> -
<a href="/css/default.php"> 新增 </a> -
<a href="/js/default.php"> 修改  </a> -
<a href="default.php"> 刪除 </a>';
?>
```

ex61-03.php
ex61-04.php

EXAMPLE 2 FOR INCLUDE STATEMENT


localhost/ex61-04.php

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

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

ex61-03.php
ex61-04.php

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:

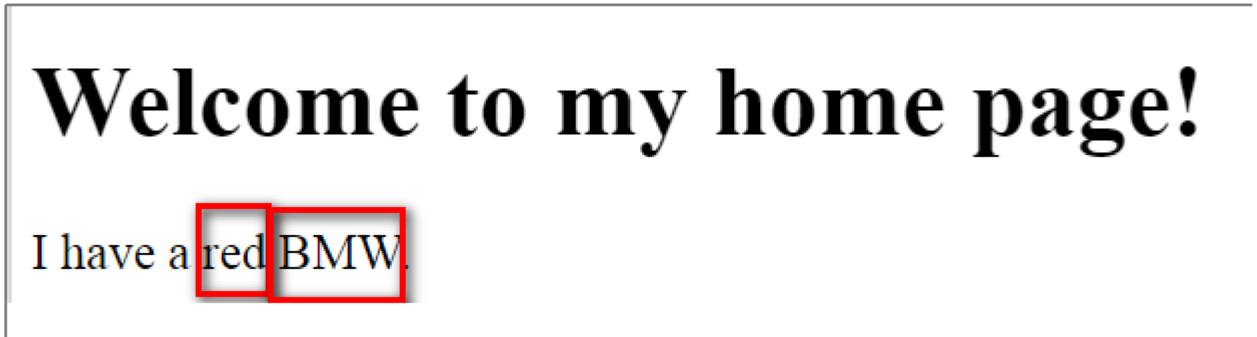
```
<?php
$color='red';
$car='BMW';
?>
```

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>
```

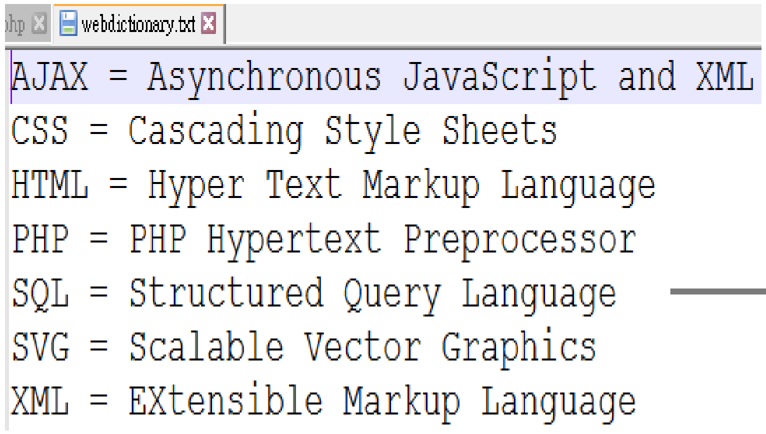


Welcome to my home page!

I have a red BMW.

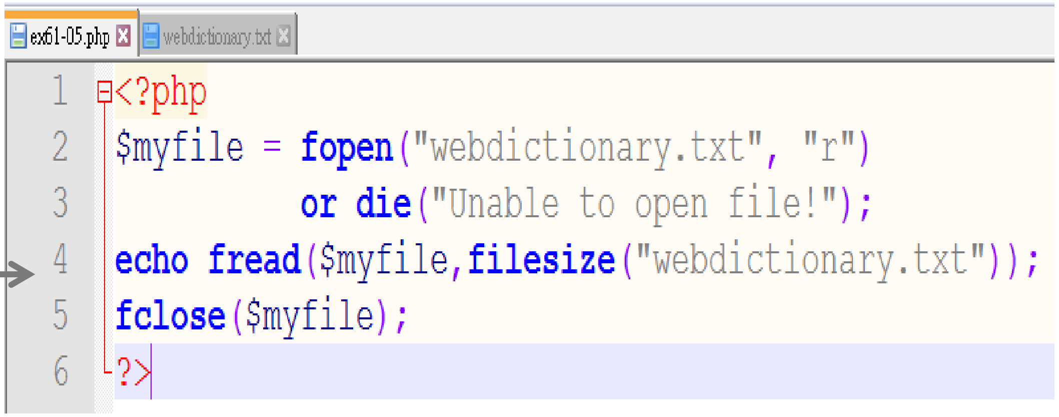
PHP FILE OPEN/READ/CLOSE

- 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

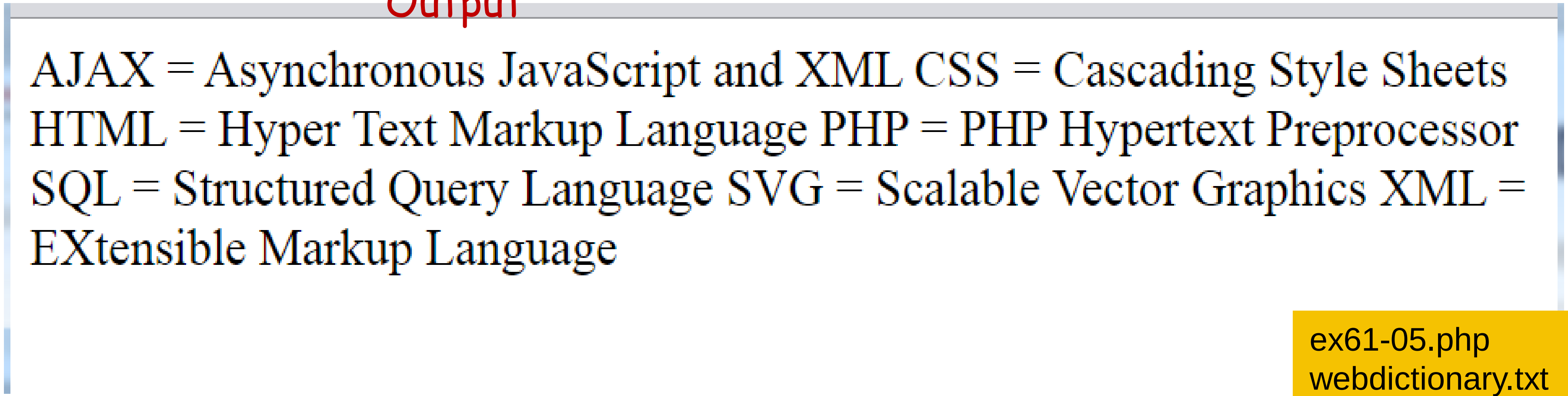
```
AJAX = Asynchronous JavaScript and XML
CSS = Cascading Style Sheets
HTML = Hyper Text Markup Language
PHP = PHP Hypertext Preprocessor
SQL = Structured Query Language
SVG = Scalable Vector Graphics
XML = EXtensible Markup Language
```



ex61-05.php

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

ex61-05.php
webdictionary.txt

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.

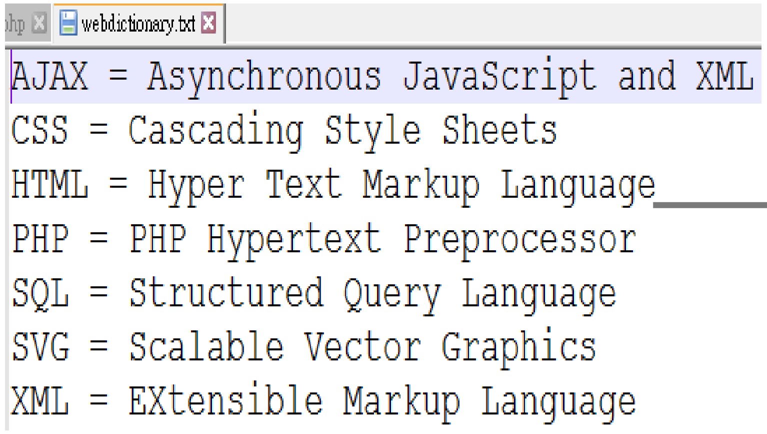
It's a good programming practice to close all files after you have finished with them. You don't want an open file running around on your server taking up resources!

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);
?>
```

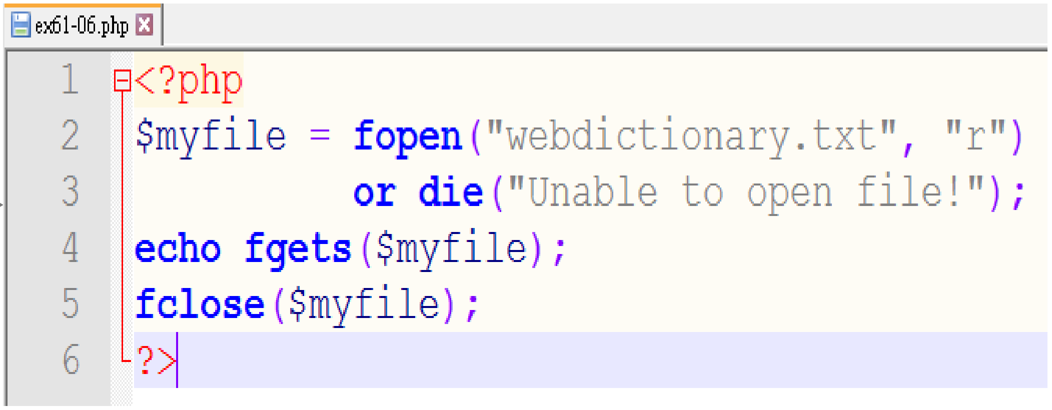
PHP READ SINGLE LINE - FGETS()

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

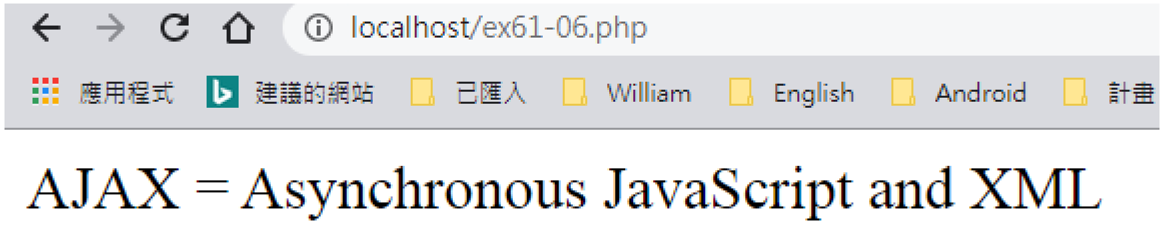
```
AJAX = Asynchronous JavaScript and XML
CSS = Cascading Style Sheets
HTML = Hyper Text Markup Language
PHP = PHP Hypertext Preprocessor
SQL = Structured Query Language
SVG = Scalable Vector Graphics
XML = EXtensible Markup Language
```



ex61-06.php

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

Output

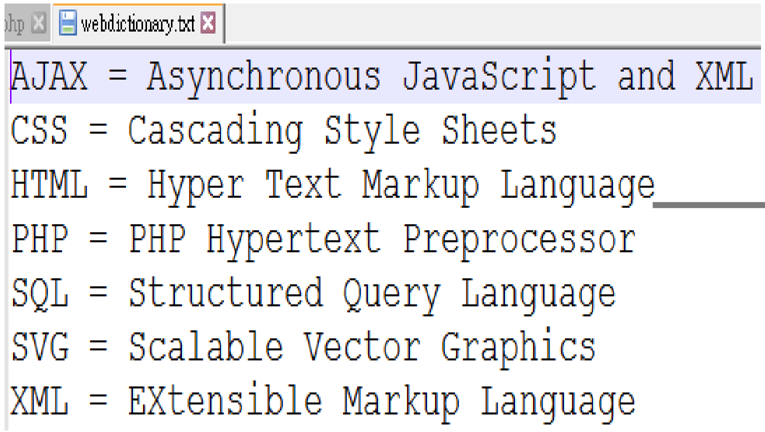


```
AJAX = Asynchronous JavaScript and XML
```

ex61-06.php
webdictionary.txt

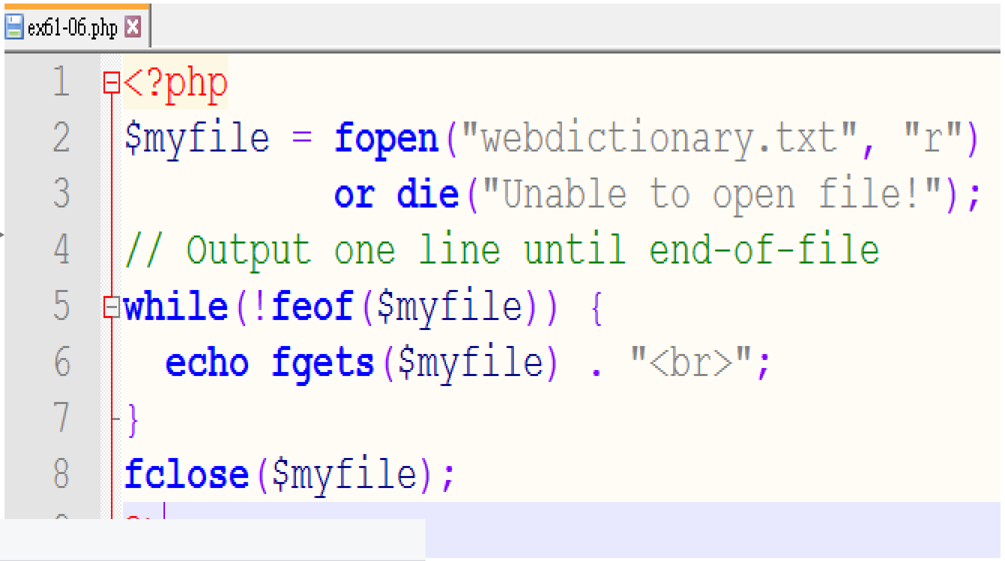
PHP CHECK END-OF-FILE - FEOF()+ FGETS()

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:



webdictionary.txt

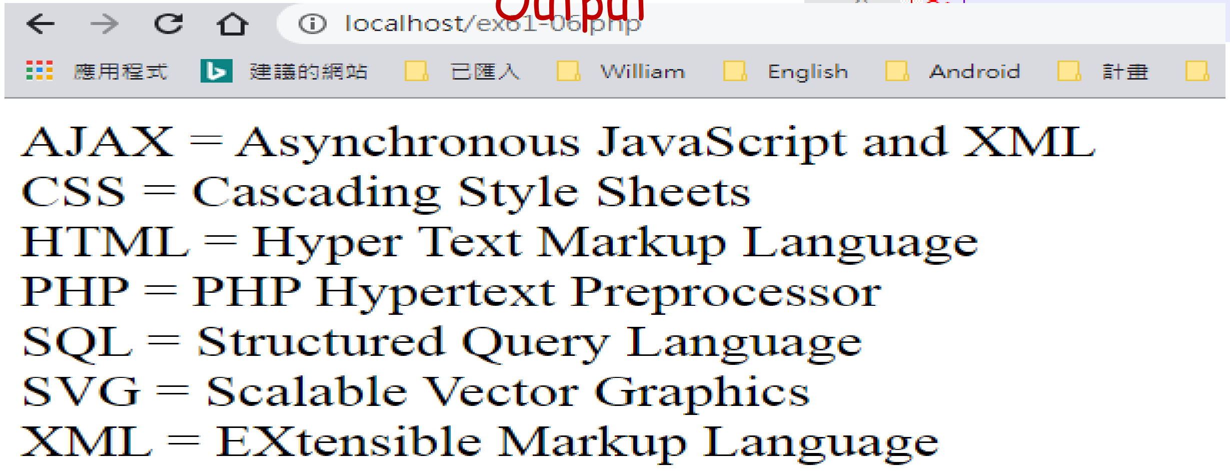
```
AJAX = Asynchronous JavaScript and XML
CSS = Cascading Style Sheets
HTML = Hyper Text Markup Language
PHP = PHP Hypertext Preprocessor
SQL = Structured Query Language
SVG = Scalable Vector Graphics
XML = EXtensible Markup Language
```



ex61-06.php

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

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
```

ex61-06.php
webdictionary.txt

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:

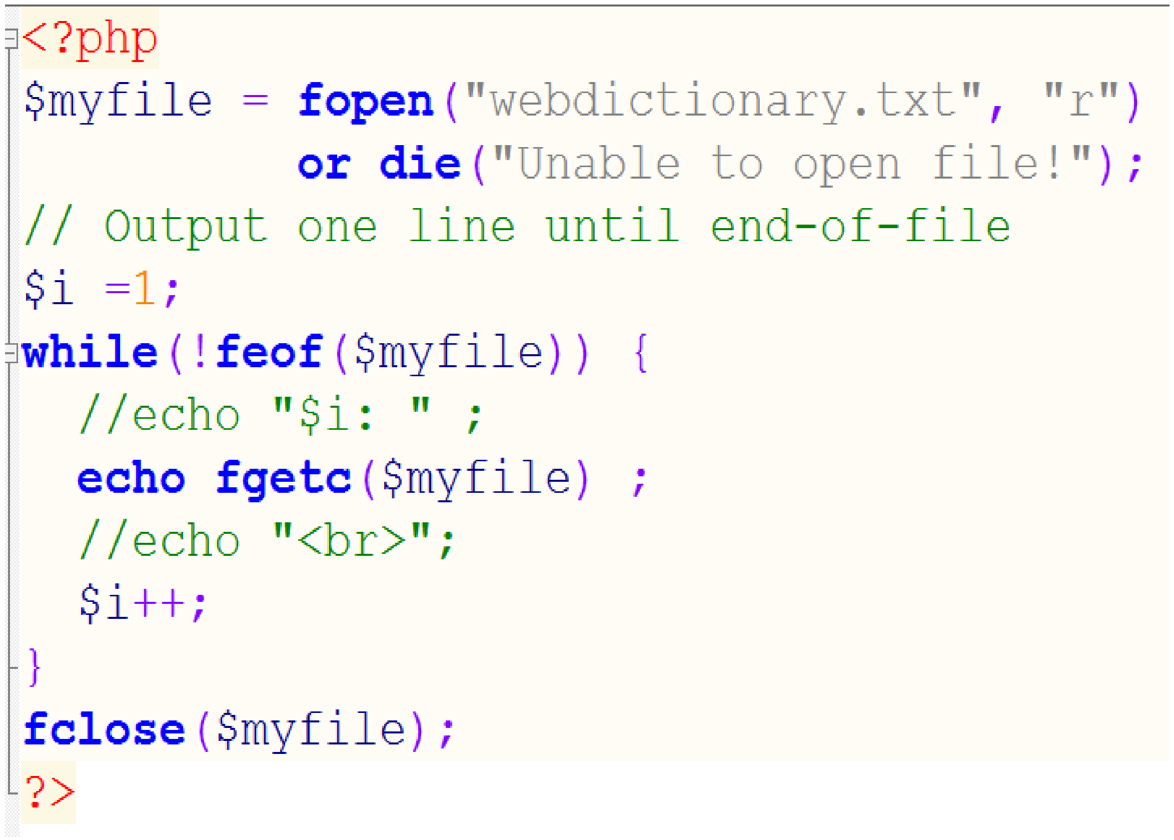
Example

```php
<?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
```

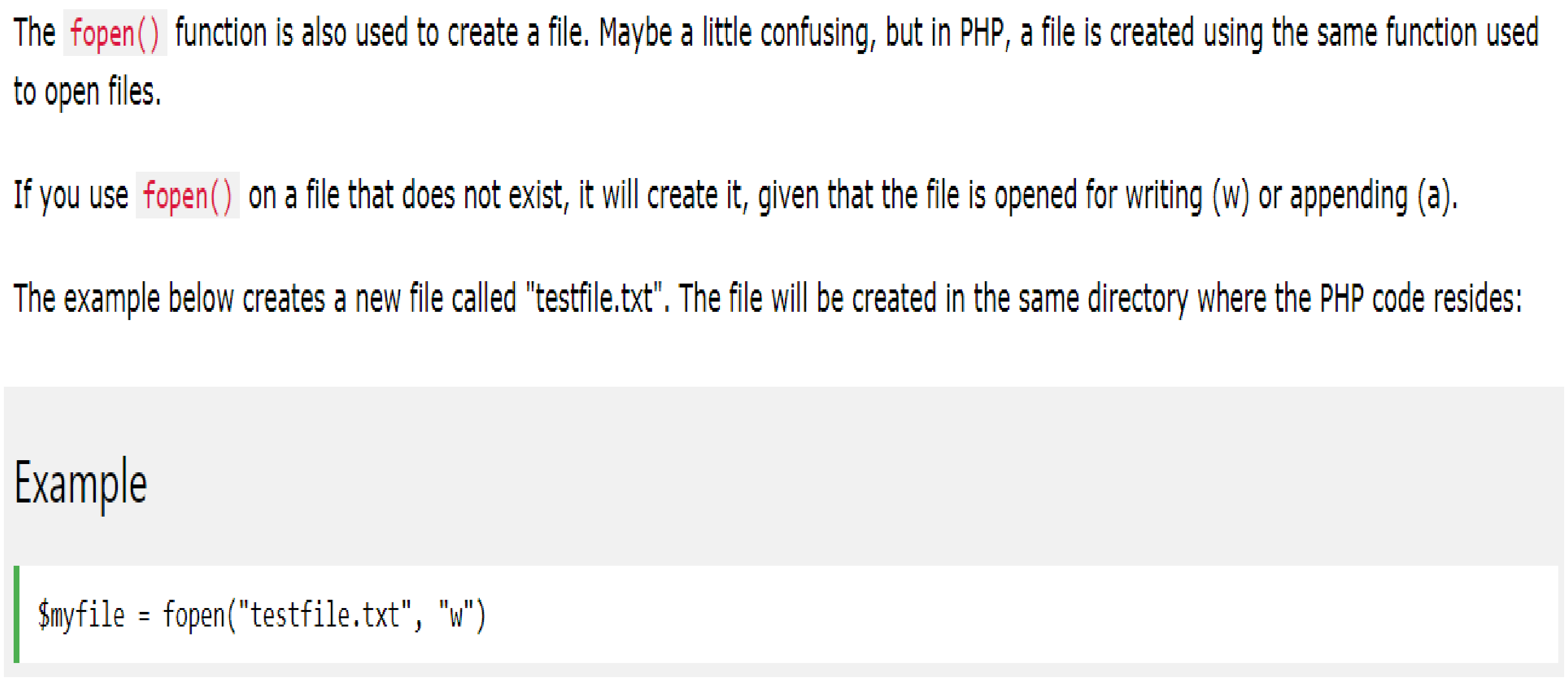
PHP READ SINGLE CHARACTER - FEOF()+ FGETC()



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

ex61-07.php
webdictionary.txt

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.

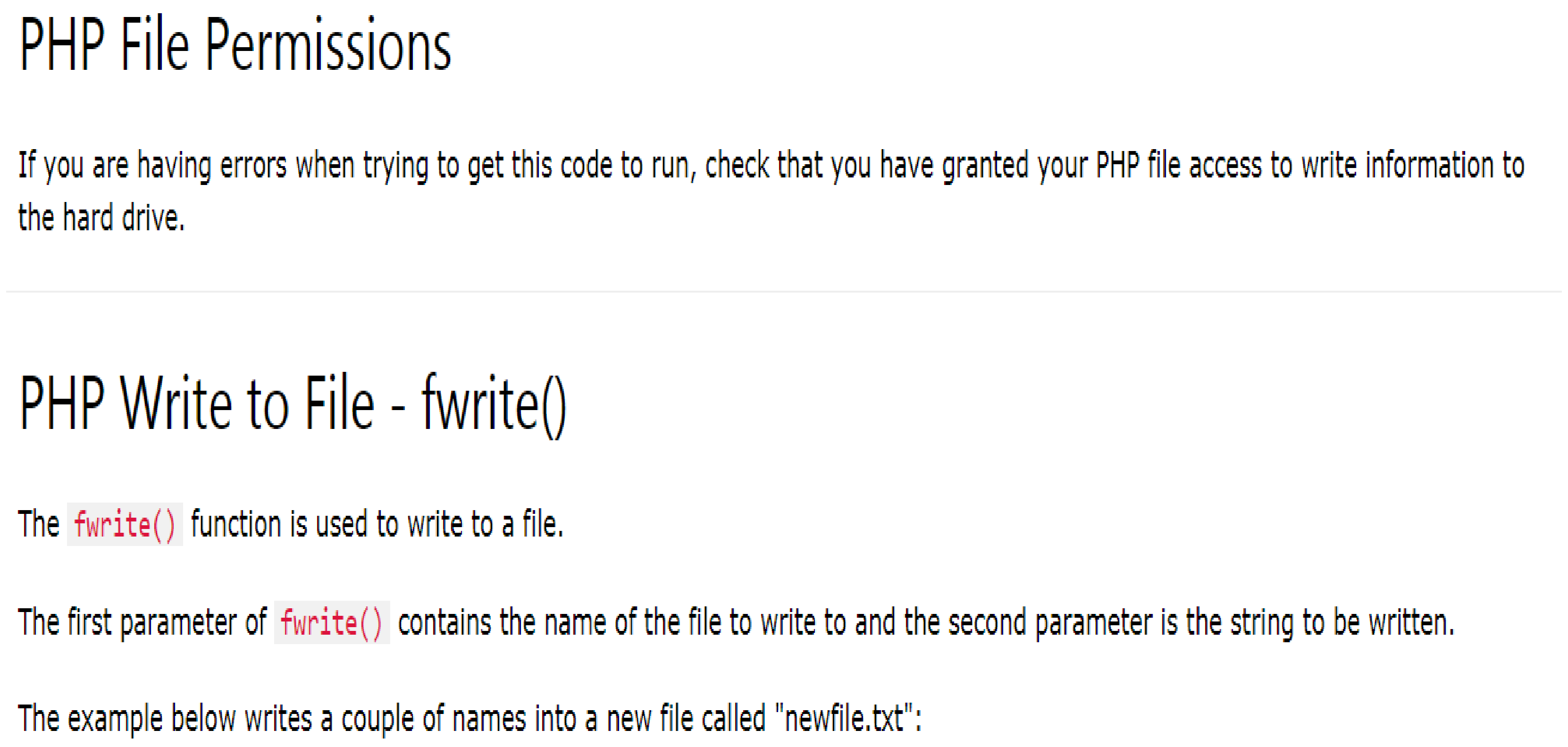
If you use `fopen()` on a file that does not exist, it will create it, given that the file is opened for writing (w) or appending (a).

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.

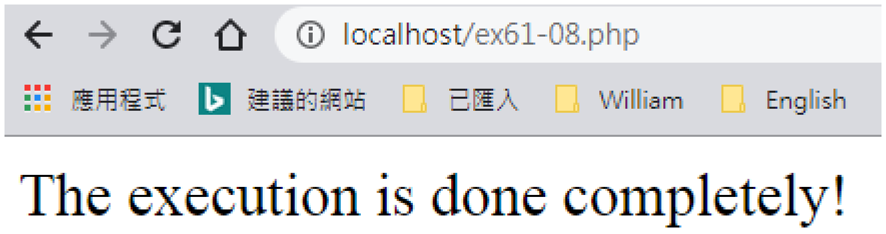
PHP Write to File - fwrite()

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

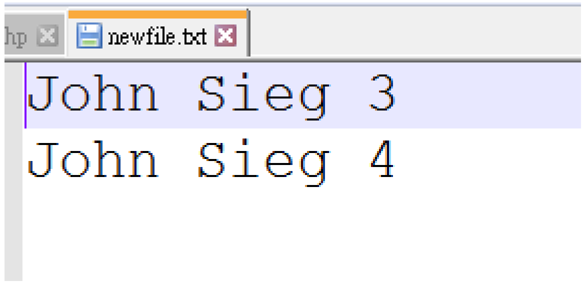
The first parameter of `fwrite()` contains the name of the file to write to and the second parameter is the string to be written.

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

PHP FOPEN-FWRITE



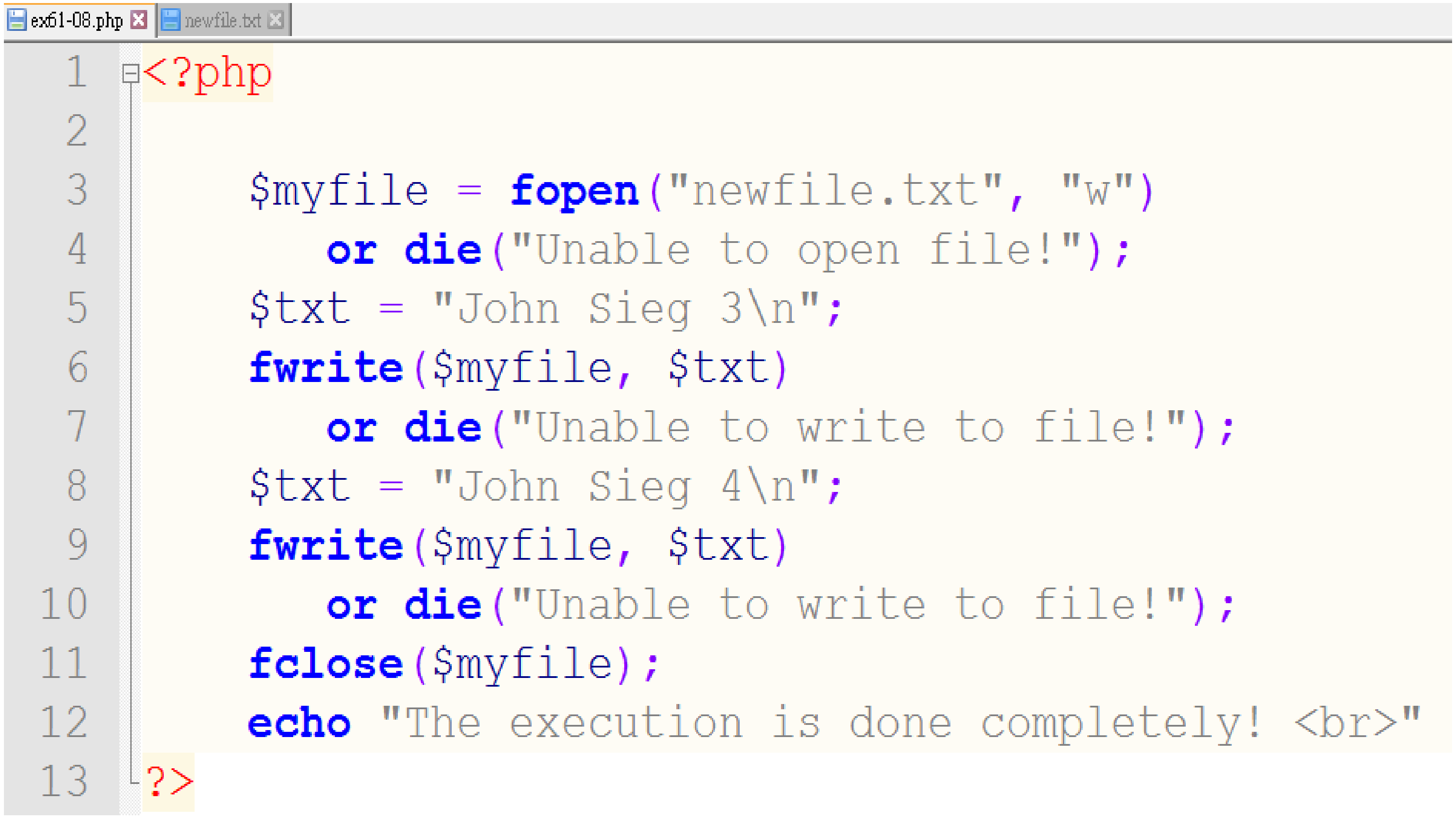
The execution is done completely!



```
John Sieg 3
John Sieg 4
```

ex61-08.php
newfile.txt

PHP FOPEN-FWRITE



ex61-08.php newfile.txt

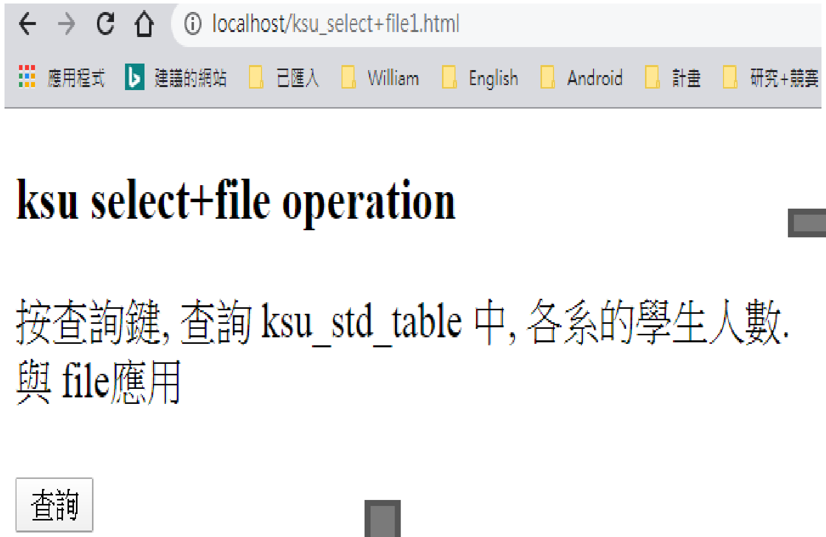
```
<?php

	$myfile = fopen("newfile.txt", "w")
		or die("Unable to open file!");
	$txt = "John Sieg 3\n";
	fwrite($myfile, $txt)
		or die("Unable to write to file!");
	$txt = "John Sieg 4\n";
	fwrite($myfile, $txt)
		or die("Unable to write to file!");
	fclose($myfile);
	echo "The execution is done completely! <br>"
?>
```

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

ex61-08.php
newfile.txt

MORE TALKS ABOUT FILES AND SOMETHING

EXERCISE: IMPLEMENT SQL SELECT AND FILE



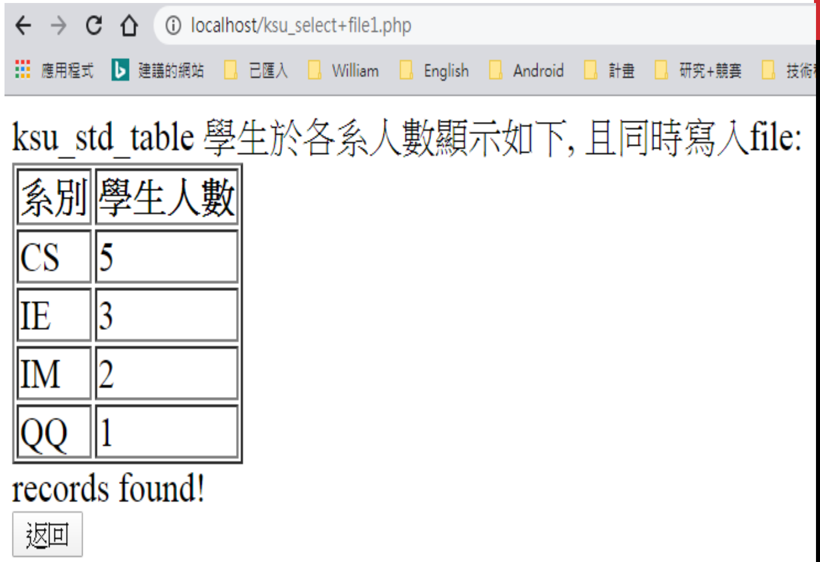
localhost/ksu_select+file1.html

應用程式 建議的網站 已匯入 William English Android 計畫 研究+競賽

ksu select+file operation

按查詢鍵, 查詢 ksu_std_table 中, 各系的學生人數. 與 file應用

查詢



localhost/ksu_select+file1.php

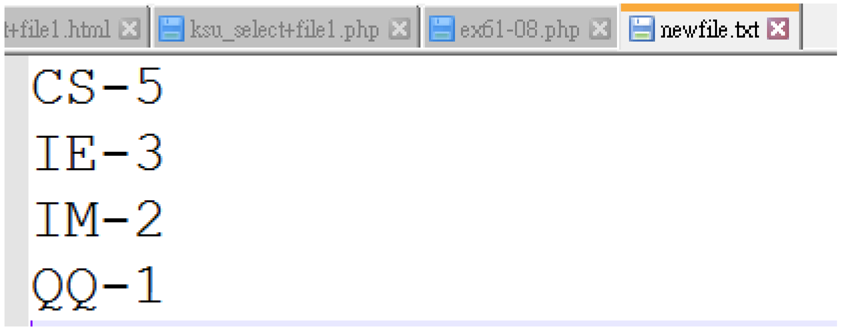
應用程式 建議的網站 已匯入 William English Android 計畫 研究+競賽 技術

ksu_std_table 學生於各系人數顯示如下, 且同時寫入file:

系別	學生人數
CS	5
IE	3
IM	2
QQ	1

records found!

返回

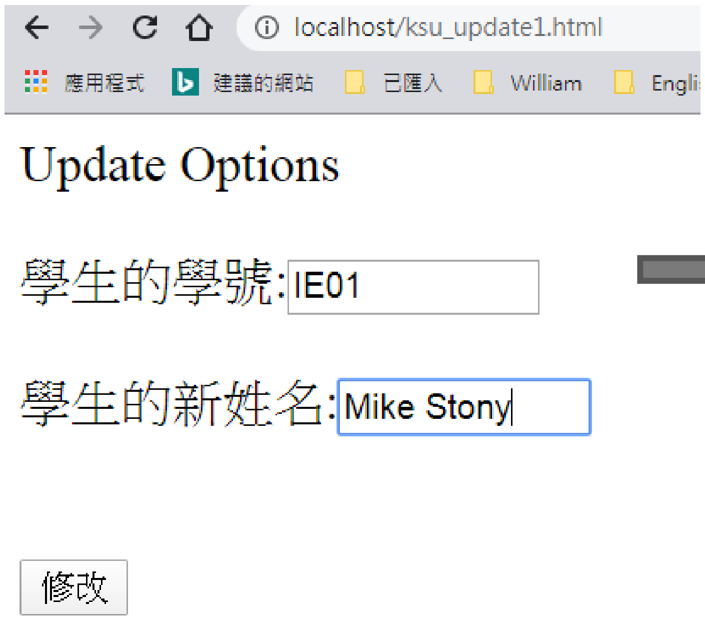


t+file1.html ksu_select+file1.php ex61-08.php newfile.txt

```
CS-5
IE-3
IM-2
QQ-1
```

ksu_select+file1.php
ksu_select+file1.html

EXERCISE: IMPLEMENT SQL UPDATE

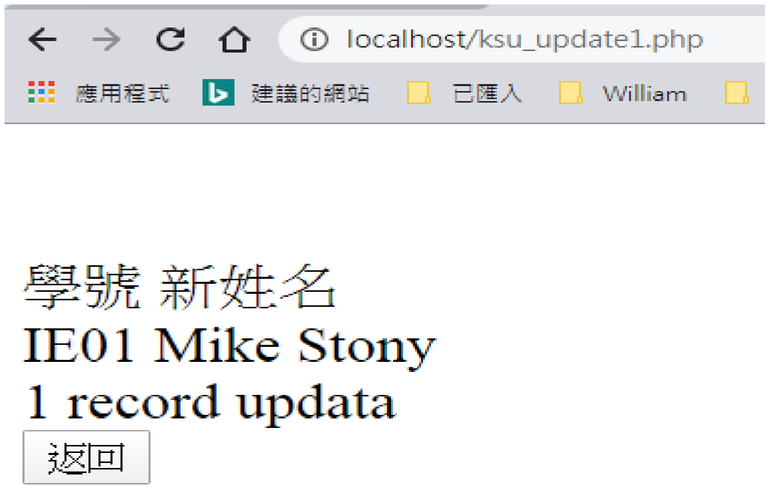


Update Options

學生的學號:IE01

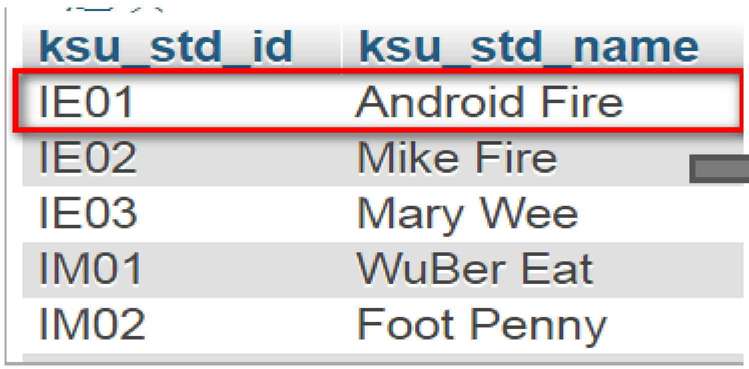
學生的新姓名:Mike Stony

修改

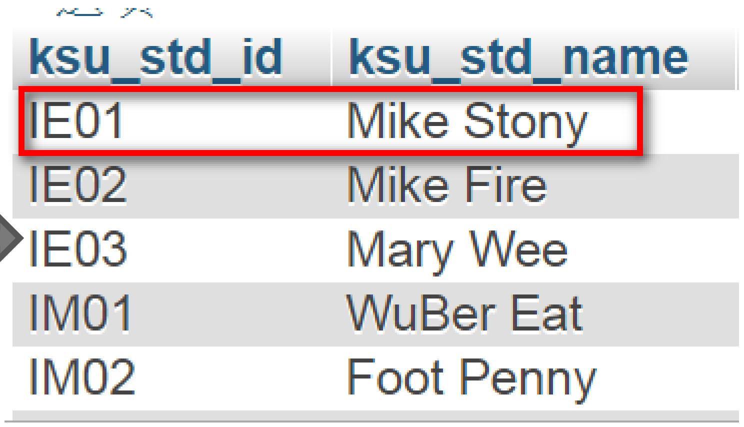


學號 新姓名
IE01 Mike Stony
1 record updata

返回



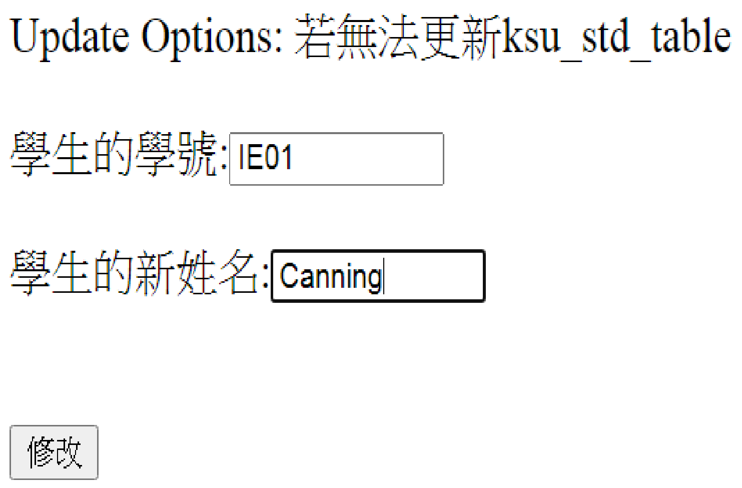
ksu_std_id	ksu_std_name
IE01	Android Fire
IE02	Mike Fire
IE03	Mary Wee
IM01	WuBer Eat
IM02	Foot Penny



ksu_std_id	ksu_std_name
IE01	Mike Stony
IE02	Mike Fire
IE03	Mary Wee
IM01	WuBer Eat
IM02	Foot Penny

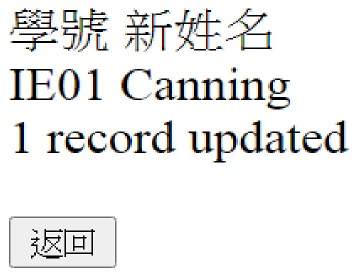
ksu_update1.html
ksu_update1.php

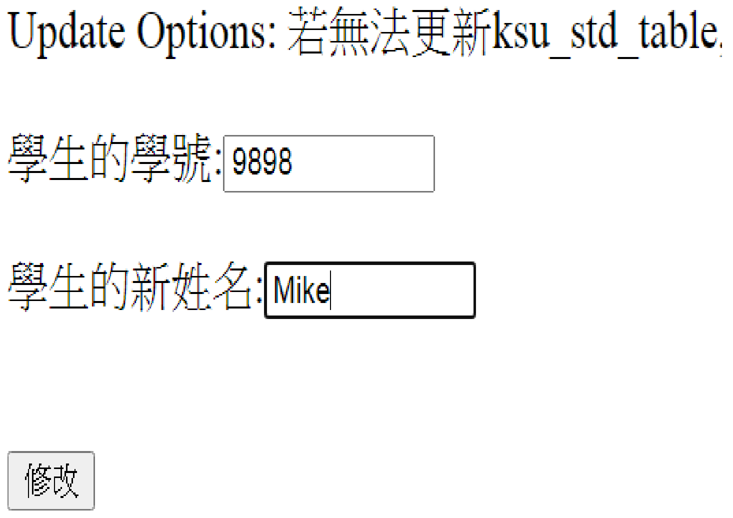
EXERCISE: IMPLEMENT SQL UPDATE+INSERT


Update Options: 若無法更新ksu_std_table

學生的學號: IE01

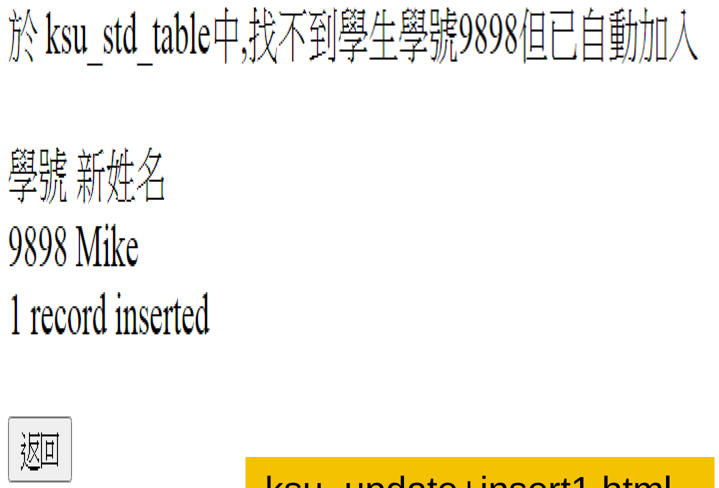
學生的新姓名: Canning


學號 新姓名
IE01 Canning
1 record updated


Update Options: 若無法更新ksu_std_table,

學生的學號: 9898

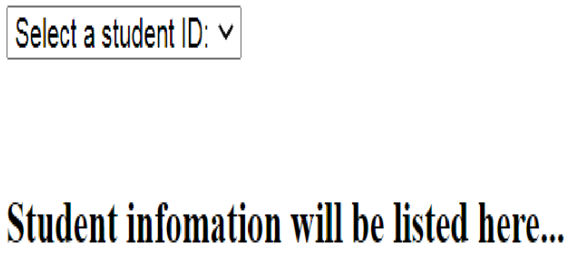
學生的新姓名: Mike


於 ksu_std_table中,找不到學生學號9898但已自動加入

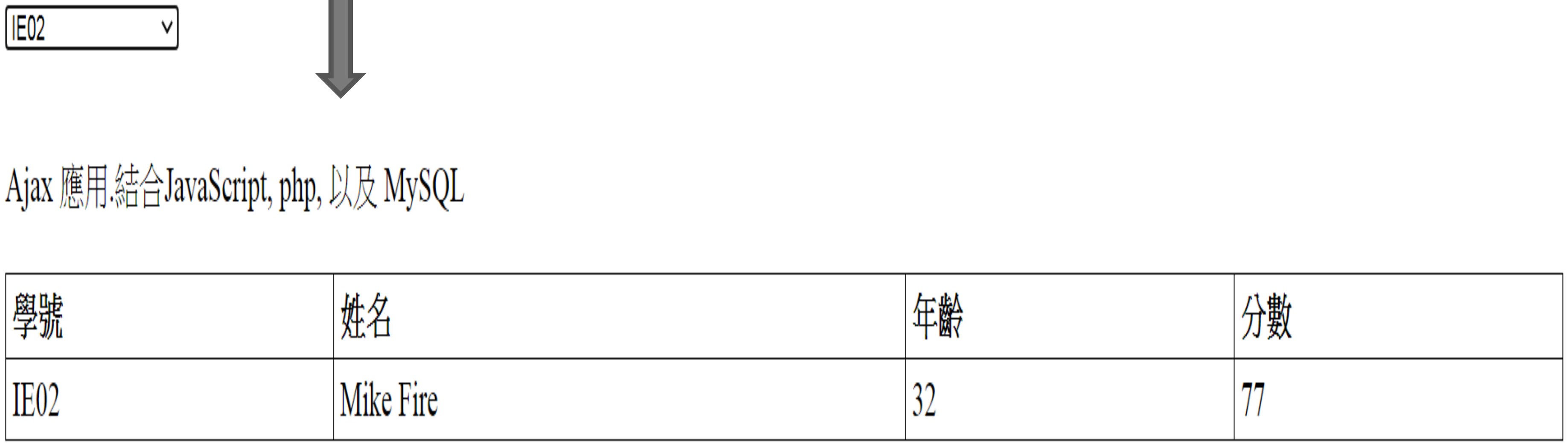
學號 新姓名
9898 Mike
1 record inserted

ksu_update+insert1.html
ksu_update+insert1.php

EXERCISE: IMPLEMENT JAVASCRIPT+PHP+MYSQL



Student infomation will be listed here...



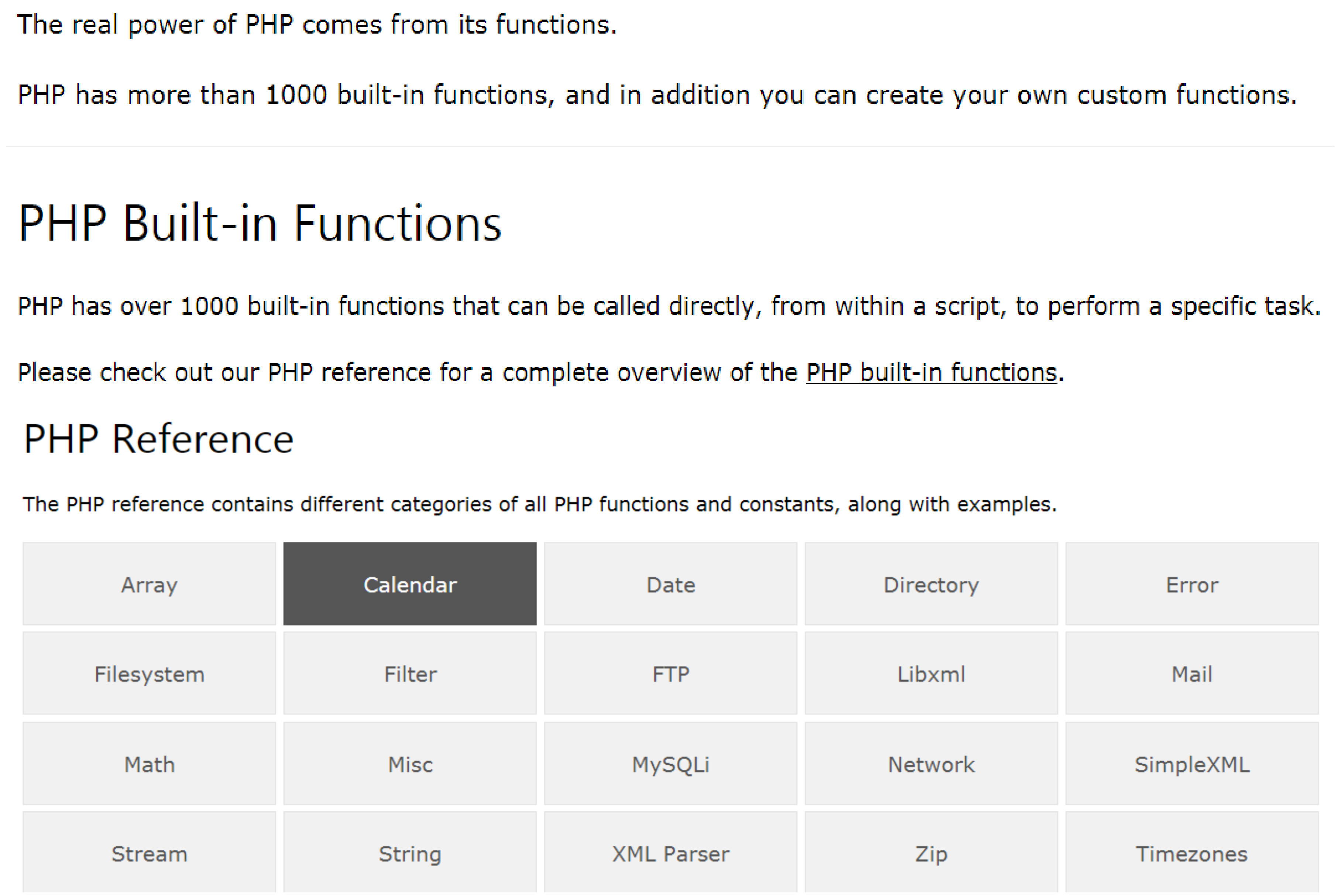
Ajax 應用.結合JavaScript, php, 以及 MySQL

學號	姓名	年齡	分數
IE02	Mike Fire	32	77

ksu_ajax_db1.html
ksu_ajax_db1.php

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.

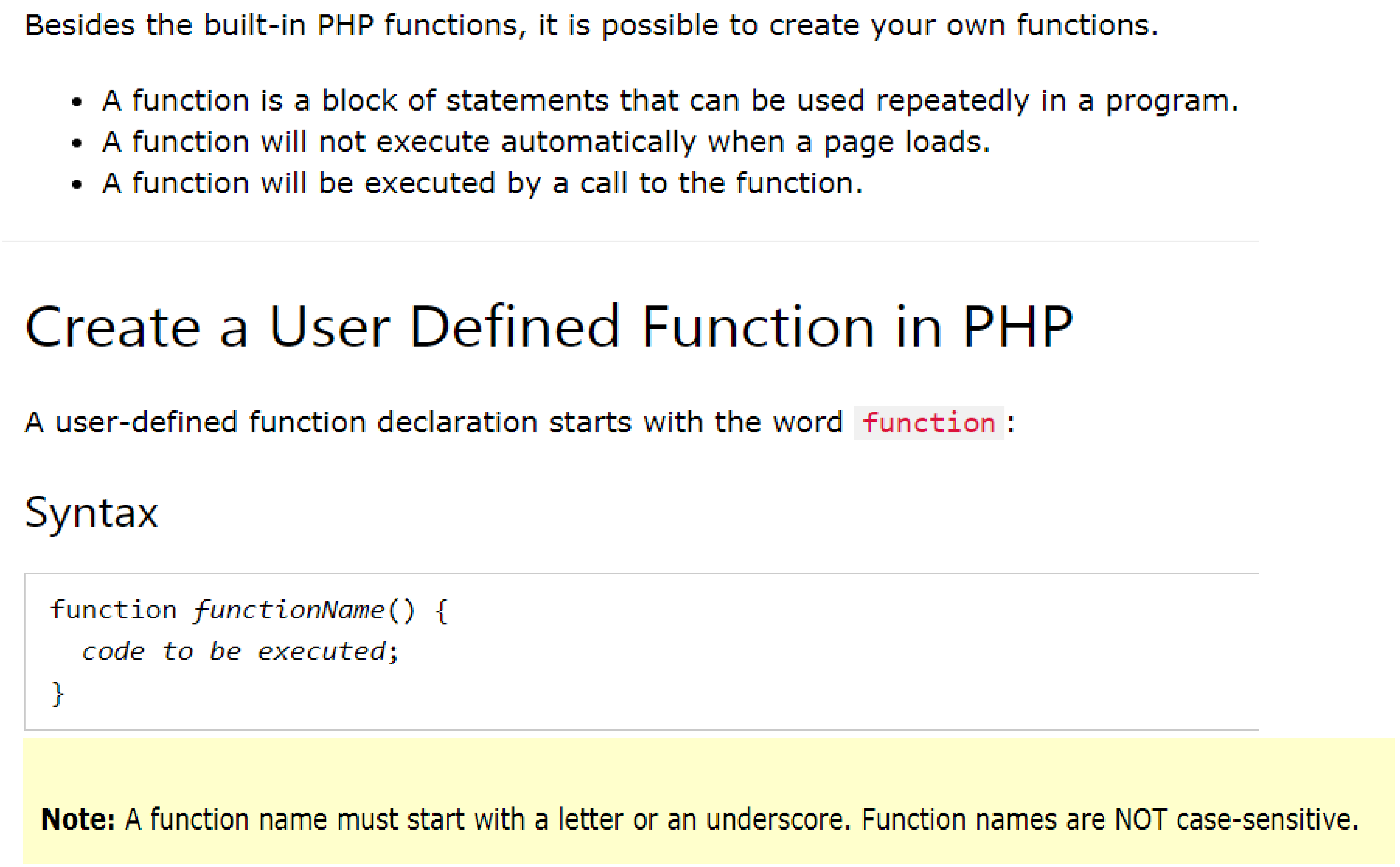
Please check out our PHP reference for a complete overview of the PHP built-in functions.

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

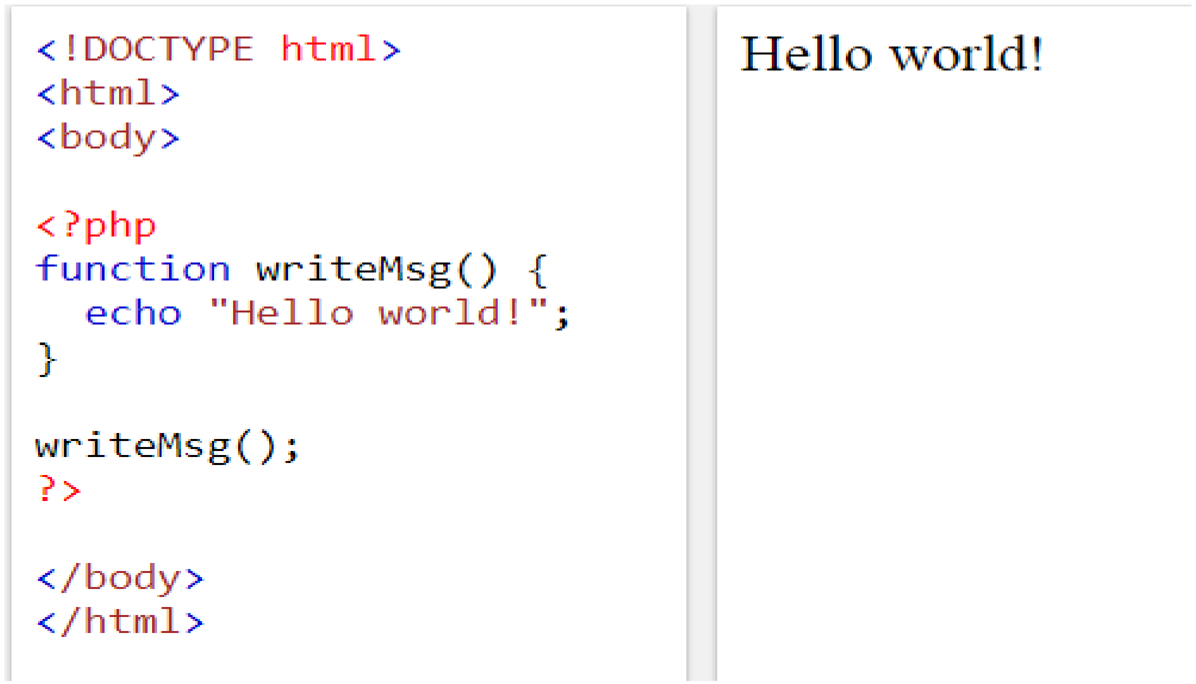
A user-defined function declaration starts with the word `function`:

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



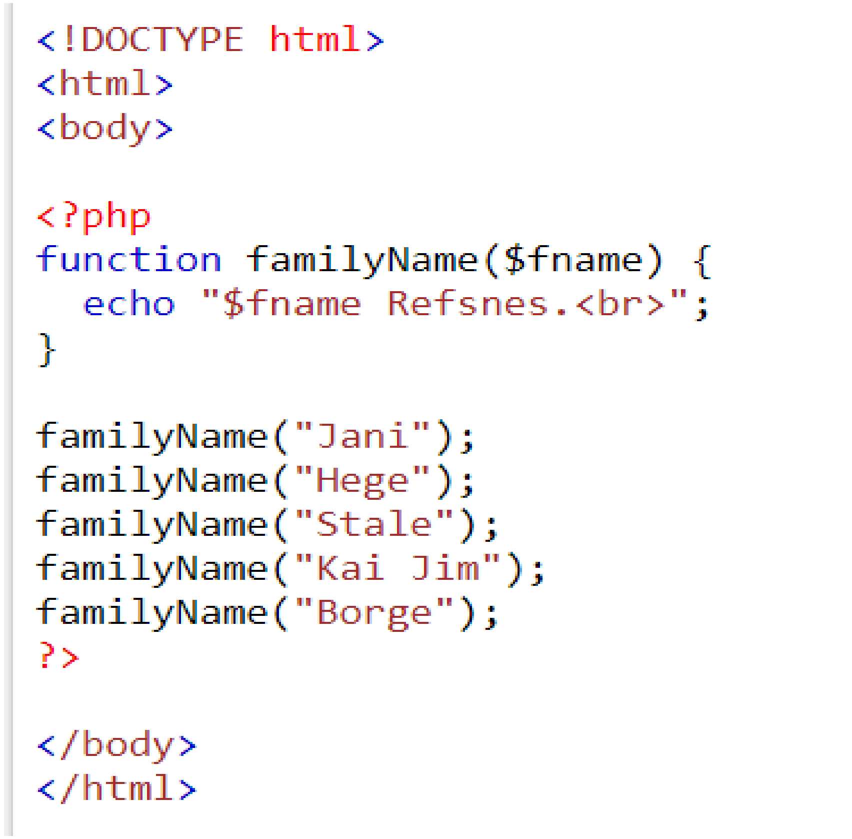
```
<!DOCTYPE html>
<html>
<body>

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

writeMsg();
?>

</body>
</html>
```

```
Hello world!
```

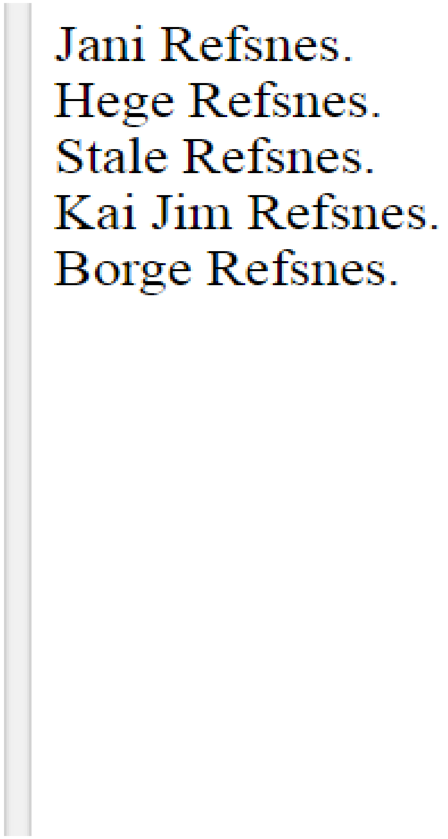
PHP USER DEFINED ARGUMENTS

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.



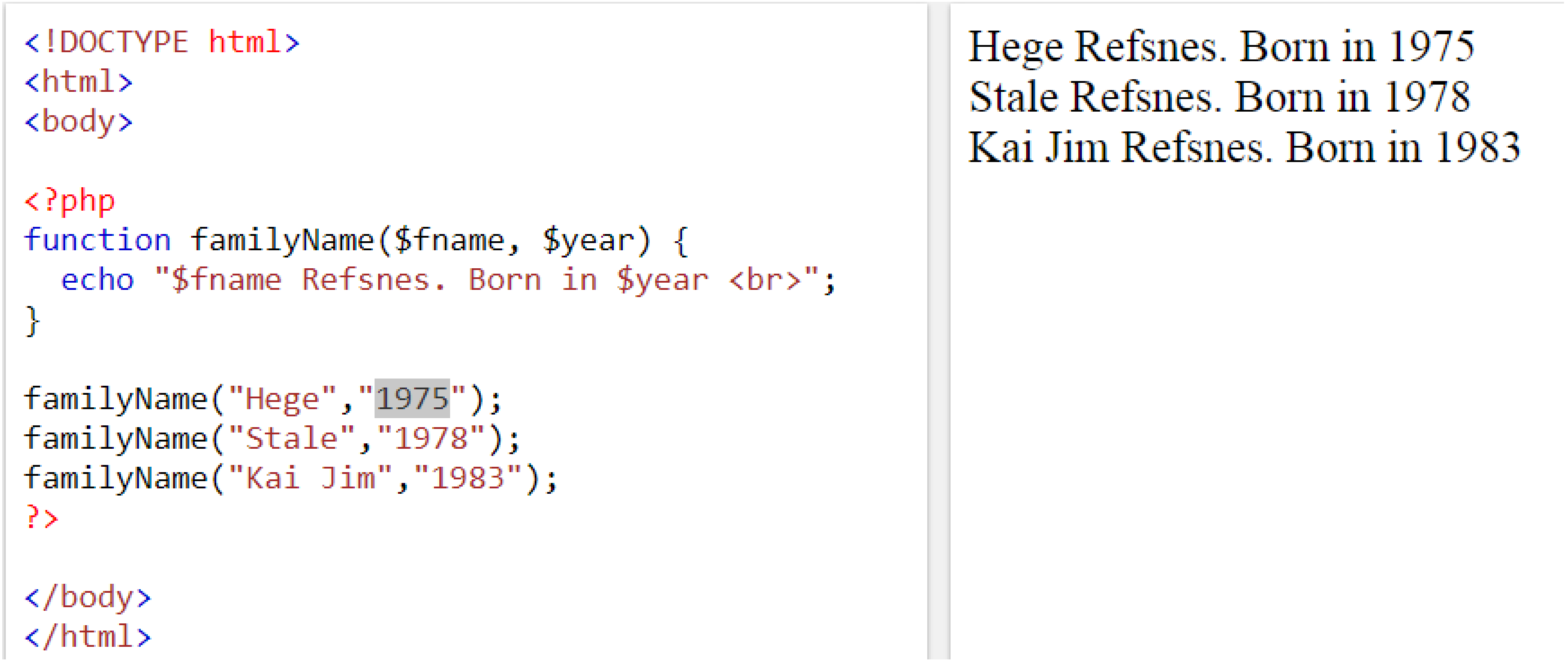
```
<!DOCTYPE html>
<html>
<body>

<?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.

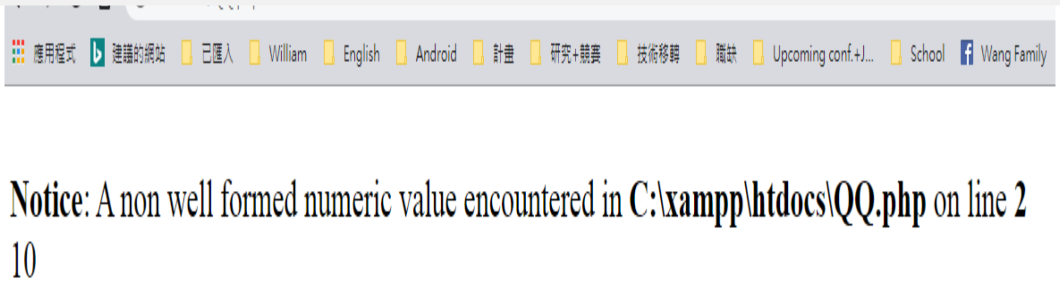
PHP automatically associates a data type to the variable, depending on its value. Since the data types are not set in a strict sense, you can do things like adding a string to an integer without causing an error.

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`:

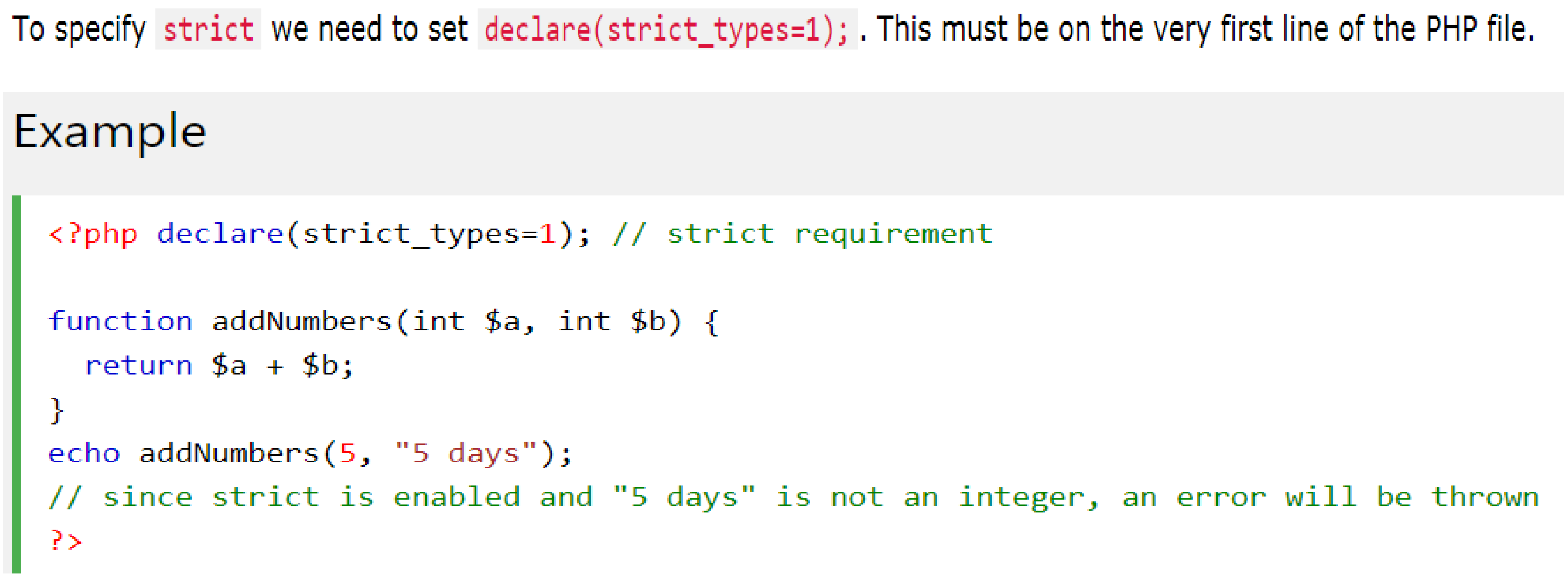
Example

```php
<?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
?>
```



```
Notice: A non well formed numeric value encountered in C:\xampp\htdocs\QQ.php on line 2
10
```

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
<?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
?>
```

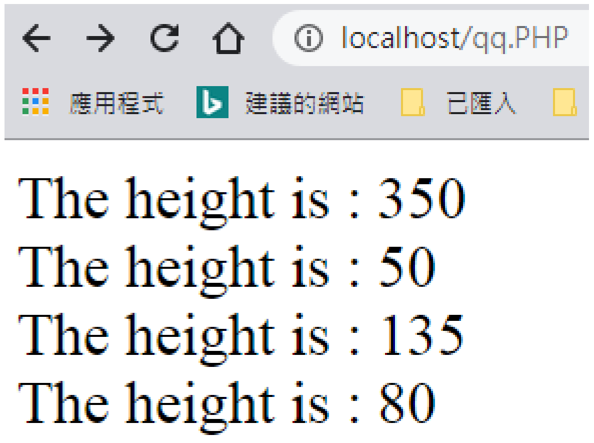
Fatal error: Uncaught TypeError: Argument 2 passed to addNumbers() must be of the type int, string given, called in C:\xampp\htdocs\QQ.php on line 6 and defined in C:\xampp\htdocs\QQ.php:3 Stack trace: #0 C:\xampp\htdocs\QQ.php(6): addNumbers(5, '5 days') #1 {main} thrown in **C:\xampp\htdocs\QQ.php**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

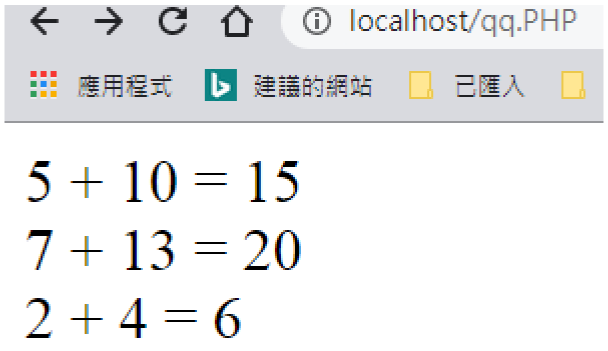


```
The height is : 350
The height is : 50
The height is : 135
The height is : 80
```

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
```

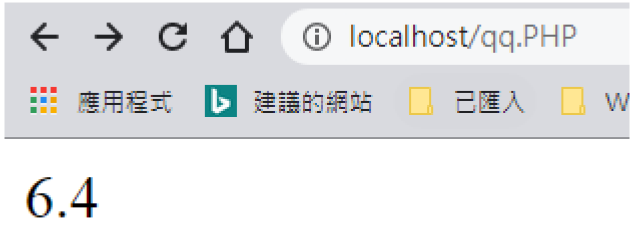
PHP RETURN TYPE DECLARATIONS

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.

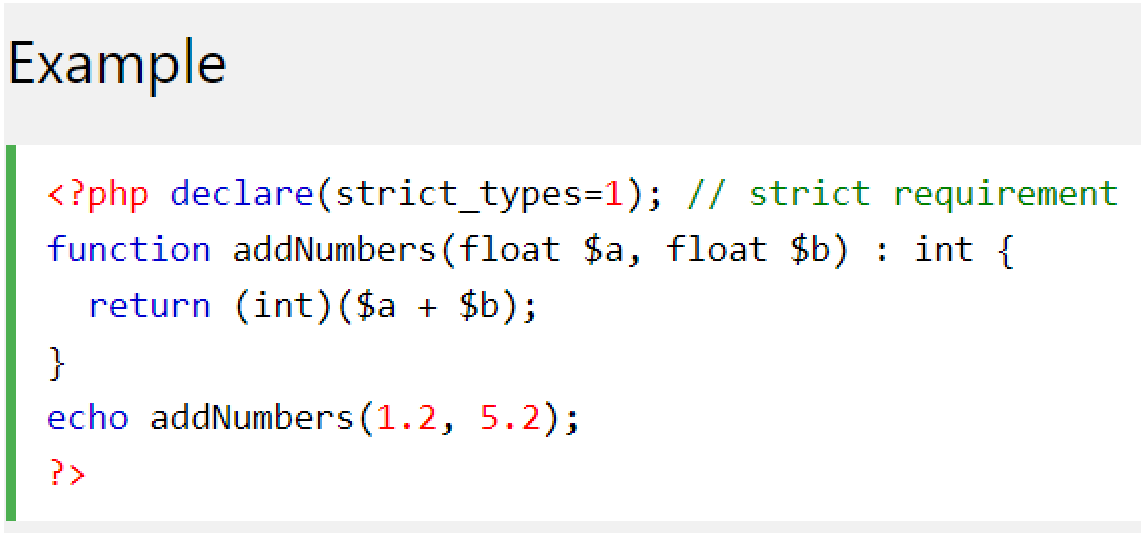
Example

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



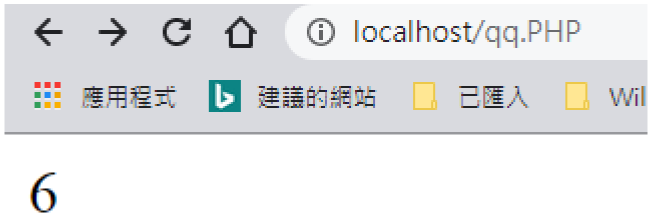
```
6.4
```

PHP RETURN TYPE DECLARATIONS



Example

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

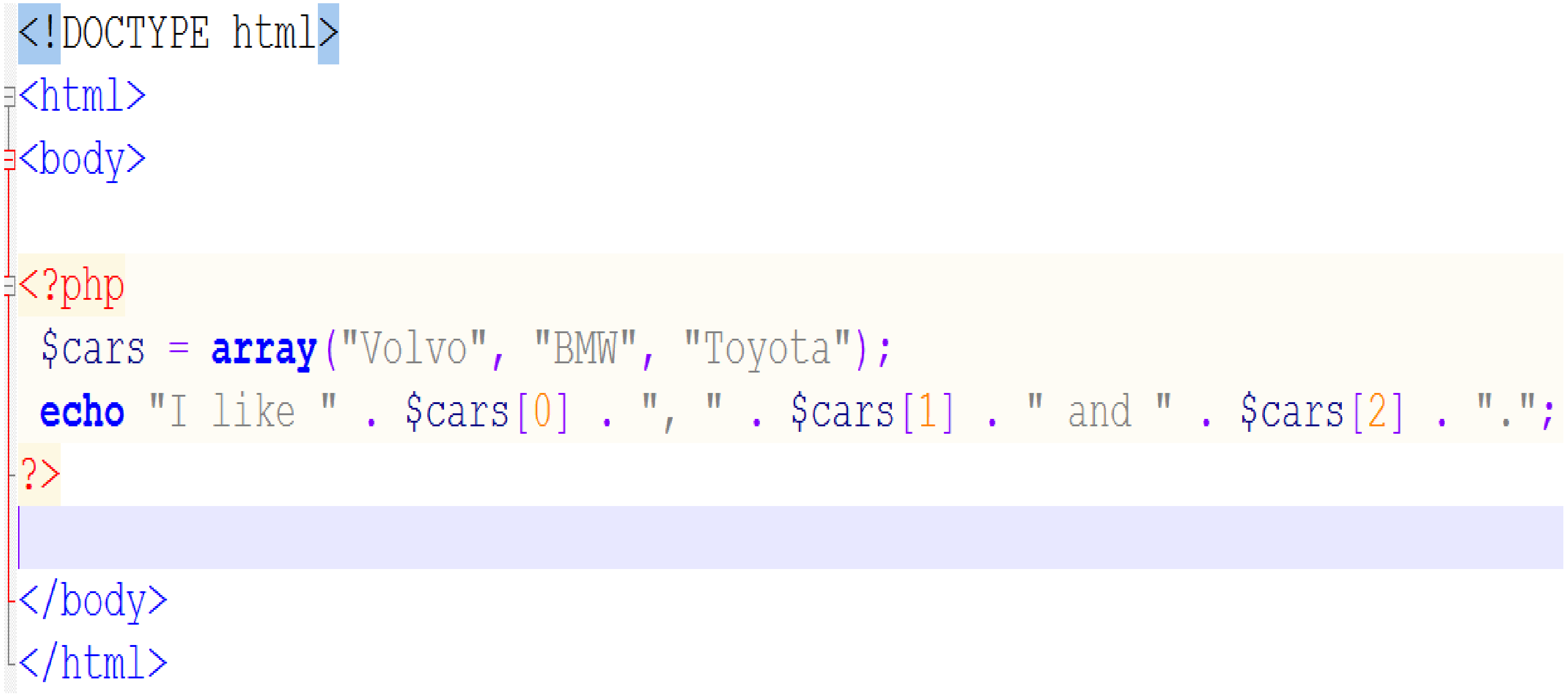


```
6
```

PHP ARRAYS

PHP ARRAY

- An array stores multiple values in one single variable.
- The following code is embedded in .php or .html.



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

<?php
 $cars = array("Volvo", "BMW", "Toyota");
 echo "I like " . $cars[0] . ", " . $cars[1] . " and " . $cars[2] . ".";
?>

</body>
</html>
```

WHAT IS AN ARRAY?

- An array is a special variable, which can hold more than one value at a time.
- If you have a list of items (a list of car names, for example), storing the cars in single variables could look like this:

```
$cars1 = "Volvo";
$cars2 = "BMW";
$cars3 = "Toyota";
```

- However, what if you want to loop through the cars and find a specific one? And what if you had not 3 cars, but 300?

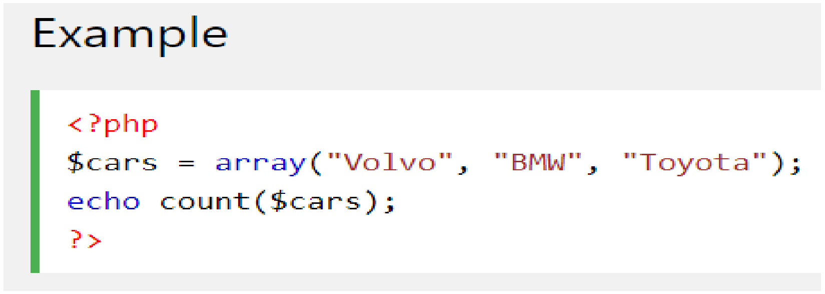
  The solution is to create an array!
- An array can hold many values under a single name, and you can access the values by referring to an index number.

CREATE AN ARRAY IN PHP

- In PHP, the **array()** function is used to create an array:

```
array();
```

- In PHP, there are three types of arrays:
  - Indexed arrays - Arrays with a numeric index
  - Associative arrays - Arrays with named keys
  - Multidimensional arrays - Arrays containing one or more arrays
- The **count()** function is used to return the length (the number of elements) of an array:



Example

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

CREATE AN ARRAY IN PHP

- In PHP, the **array()** function is used to create an array:

```
array();
```

- In PHP, there are three types of arrays:
  - Indexed arrays - Arrays with a numeric index
  - Associative arrays - Arrays with named keys
  - Multidimensional arrays - Arrays containing one or more arrays
- The **count()** function is used to return the length (the number of elements) of an array:

PHP INDEXED ARRAYS

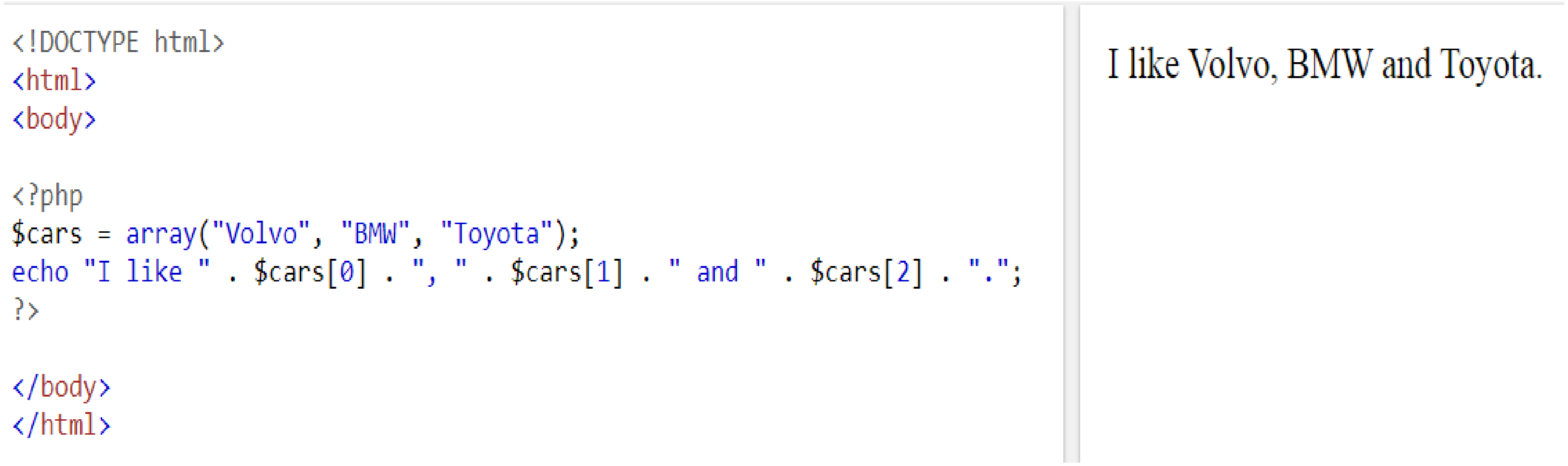
- There are two ways to create indexed arrays:
  - The index can be assigned automatically (index always starts at 0), like this:

    ```
    $cars = array("Volvo", "BMW", "Toyota");
    ```

  - or the index can be assigned manually:

    ```
    $cars[0] = "Volvo";
    $cars[1] = "BMW";
    $cars[2] = "Toyota";
    ```

  - For example,

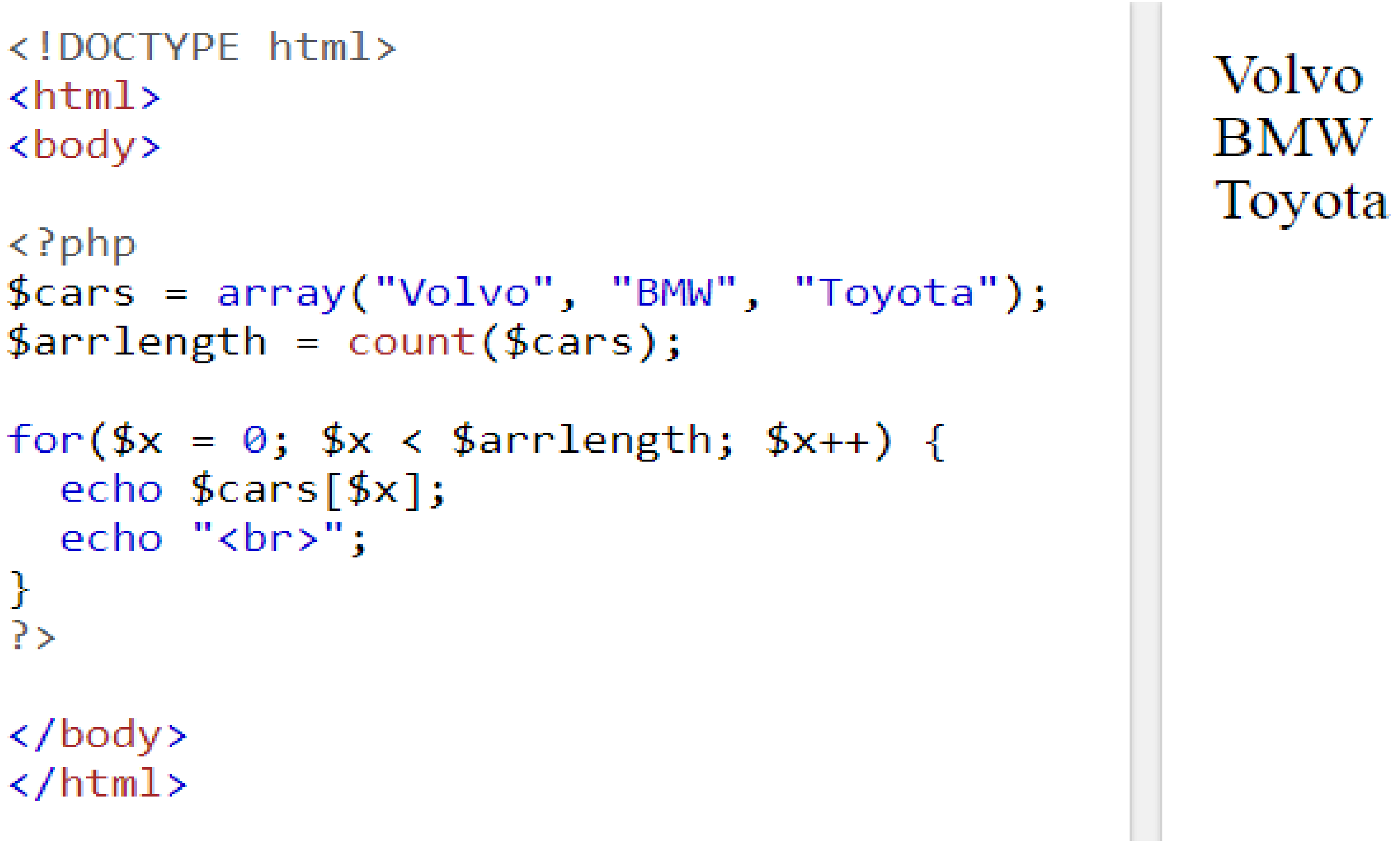


```
<!DOCTYPE html>
<html>
<body>

<?php
$cars = array("Volvo", "BMW", "Toyota");
echo "I like " . $cars[0] . ", " . $cars[1] . " and " . $cars[2] . ".";
?>

</body>
</html>
```

```
I like Volvo, BMW and Toyota.
```

LOOP THROUGH AN INDEXED ARRAY

- To loop through and print all the values of an indexed array, you could use a **for** loop, like this:
- For instance,



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

<?php
$cars = array("Volvo", "BMW", "Toyota");
$arrlength = count($cars);

for($x = 0; $x < $arrlength; $x++) {
  echo $cars[$x];
  echo "<br>";
}
?>

</body>
</html>
```

```
Volvo
BMW
Toyota
```

PHP ASSOCIATIVE ARRAYS

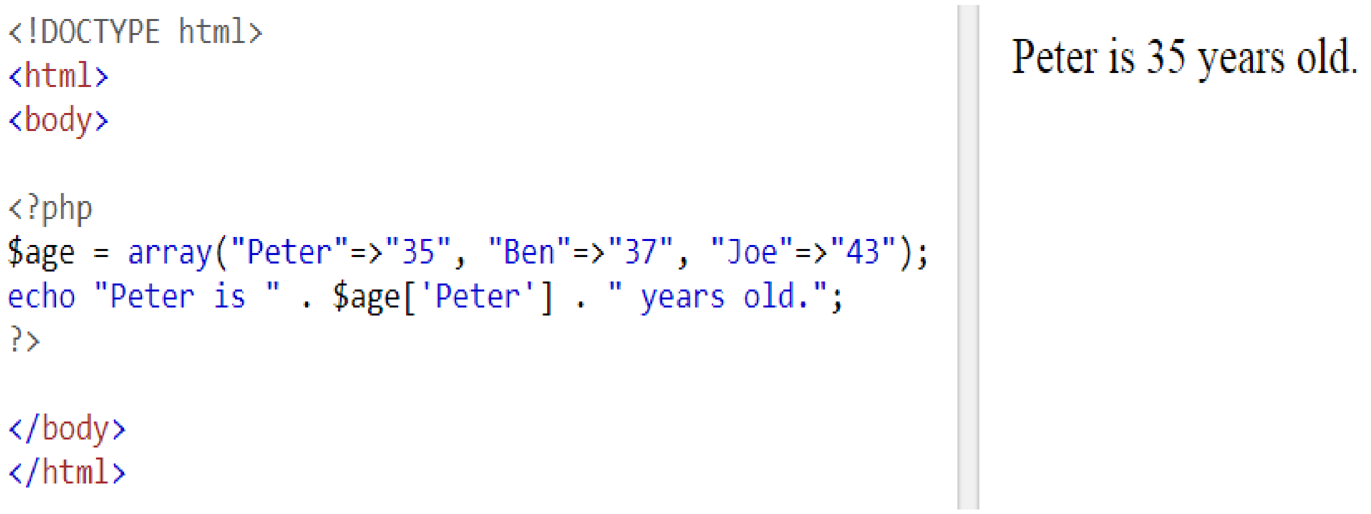
- Associative arrays are arrays that use named keys that you assign to them.
- There are two ways to create an associative array:

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

- or, the named keys can then be used in a script:

```
$age['Peter'] = "35";
$age['Ben'] = "37";
$age['Joe'] = "43";
```

- For example,

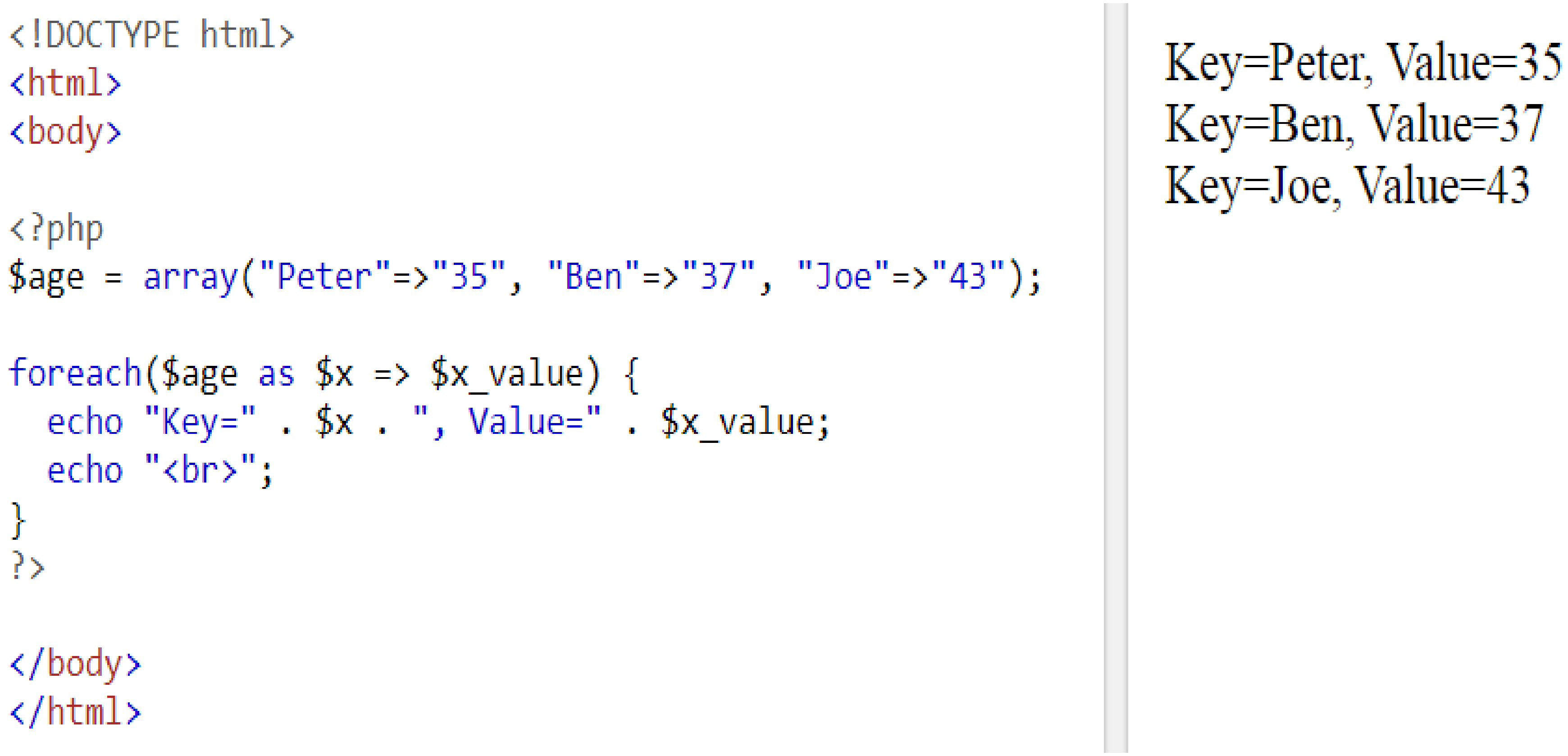


```
<!DOCTYPE html>
<html>
<body>

<?php
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");
echo "Peter is " . $age['Peter'] . " years old.";
?>

</body>
</html>
```

```
Peter is 35 years old.
```

PHP ASSOCIATIVE ARRAYS

- To loop through and print all the values of an associative array, you could use a **foreach** loop, like this:

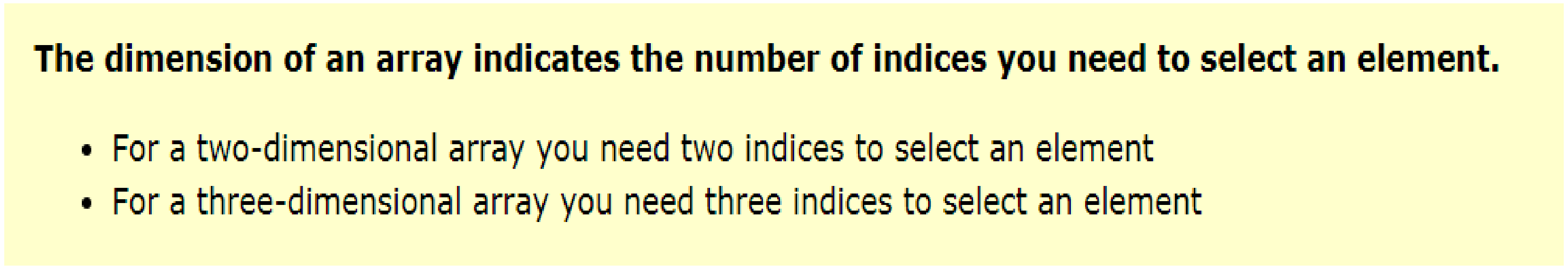


```
<!DOCTYPE html>
<html>
<body>

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

foreach($age as $x => $x_value) {
  echo "Key=" . $x . ", Value=" . $x_value;
  echo "<br>";
}
?>

</body>
</html>
```

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

PHP - MULTIDIMENSIONAL ARRAYS

- In the previous pages, we have described arrays that are a single list of key/value pairs. However, sometimes you want to store values with more than one key. For this, we have multidimensional arrays
- A multidimensional array is an array containing one or more arrays.
- PHP supports multidimensional arrays that are two, three, four, five, or more levels deep. However, arrays more than three levels deep are hard to manage for most people.



The dimension of an array indicates the number of indices you need to select an element.

- For a two-dimensional array you need two indices to select an element
- For a three-dimensional array you need three indices to select an element

PHP - MULTIDIMENSIONAL ARRAYS

- A two-dimensional array is an array of arrays (a three-dimensional array is an array of arrays of arrays). First, take a look at the following table:

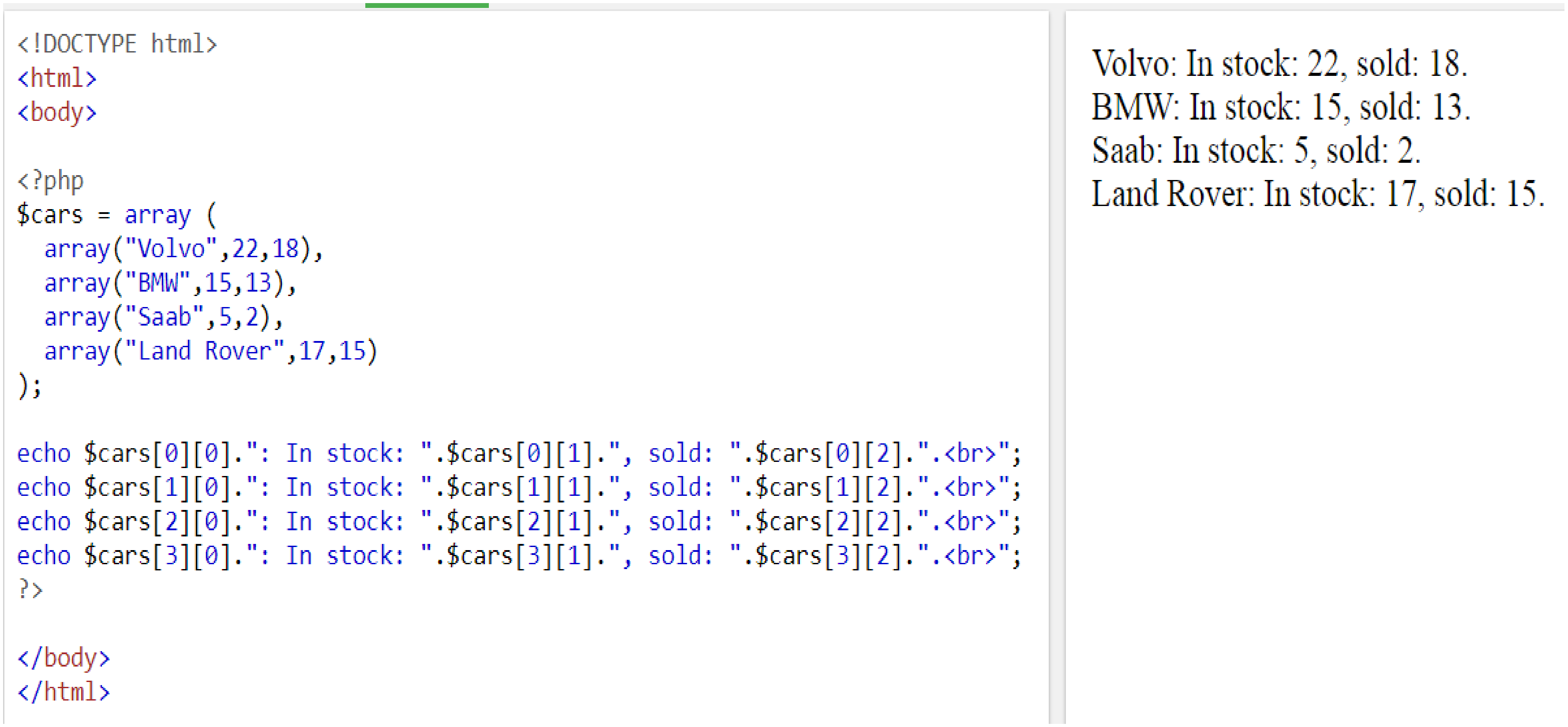
Name	Stock	Sold
Volvo	22	18
BMW	15	13
Saab	5	2
Land Rover	17	15

- We can store the data from the table above in a two-dimensional array, like this:

```
$cars = array (
  array("Volvo",22,18),
  array("BMW",15,13),
  array("Saab",5,2),
  array("Land Rover",17,15)
);
```

PHP - MULTIDIMENSIONAL ARRAYS 2/N

- Now the two-dimensional $cars array contains four arrays, and it has two indices: row and column. To get access to the elements of the $cars array we must point to the two indices (row and column):

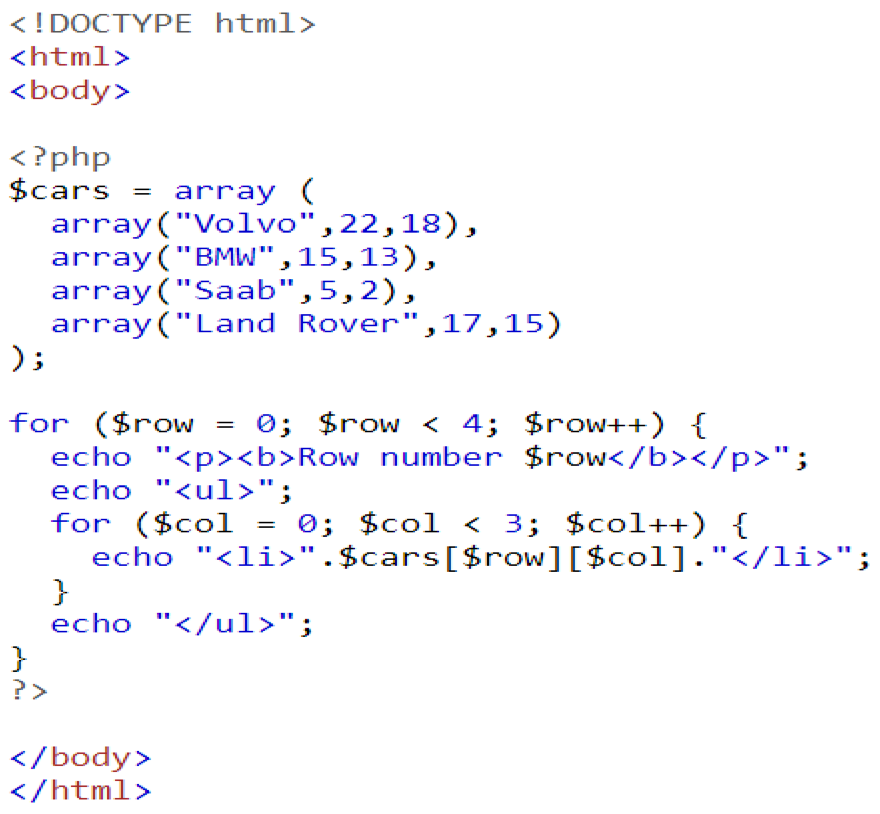


```
<!DOCTYPE html>
<html>
<body>

<?php
$cars = array (
  array("Volvo",22,18),
  array("BMW",15,13),
  array("Saab",5,2),
  array("Land Rover",17,15)
);

echo $cars[0][0].": In stock: ".$cars[0][1].", sold: ".$cars[0][2].".<br>";
echo $cars[1][0].": In stock: ".$cars[1][1].", sold: ".$cars[1][2].".<br>";
echo $cars[2][0].": In stock: ".$cars[2][1].", sold: ".$cars[2][2].".<br>";
echo $cars[3][0].": In stock: ".$cars[3][1].", sold: ".$cars[3][2].".<br>";
?>

</body>
</html>
```

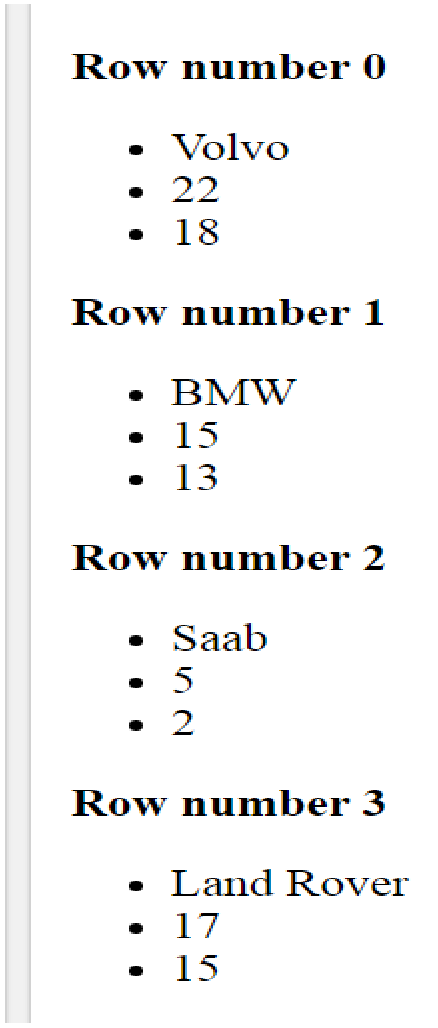
```
Volvo: In stock: 22, sold: 18.
BMW: In stock: 15, sold: 13.
Saab: In stock: 5, sold: 2.
Land Rover: In stock: 17, sold: 15.
```

PHP - MULTIDIMENSIONAL ARRAYS

- We can also put a **for** loop inside another **for** loop to get the elements of the $cars array (we still have to point to the two indices):



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

<?php
$cars = array (
  array("Volvo",22,18),
  array("BMW",15,13),
  array("Saab",5,2),
  array("Land Rover",17,15)
);

for ($row = 0; $row < 4; $row++) {
  echo "<p><b>Row number $row</b></p>";
  echo "<ul>";
  for ($col = 0; $col < 3; $col++) {
    echo "<li>".$cars[$row][$col]."</li>";
  }
  echo "</ul>";
}
?>

</body>
</html>
```



Row number 0

- Volvo
- 22
- 18

Row number 1

- BMW
- 15
- 13

Row number 2

- Saab
- 5
- 2

Row number 3

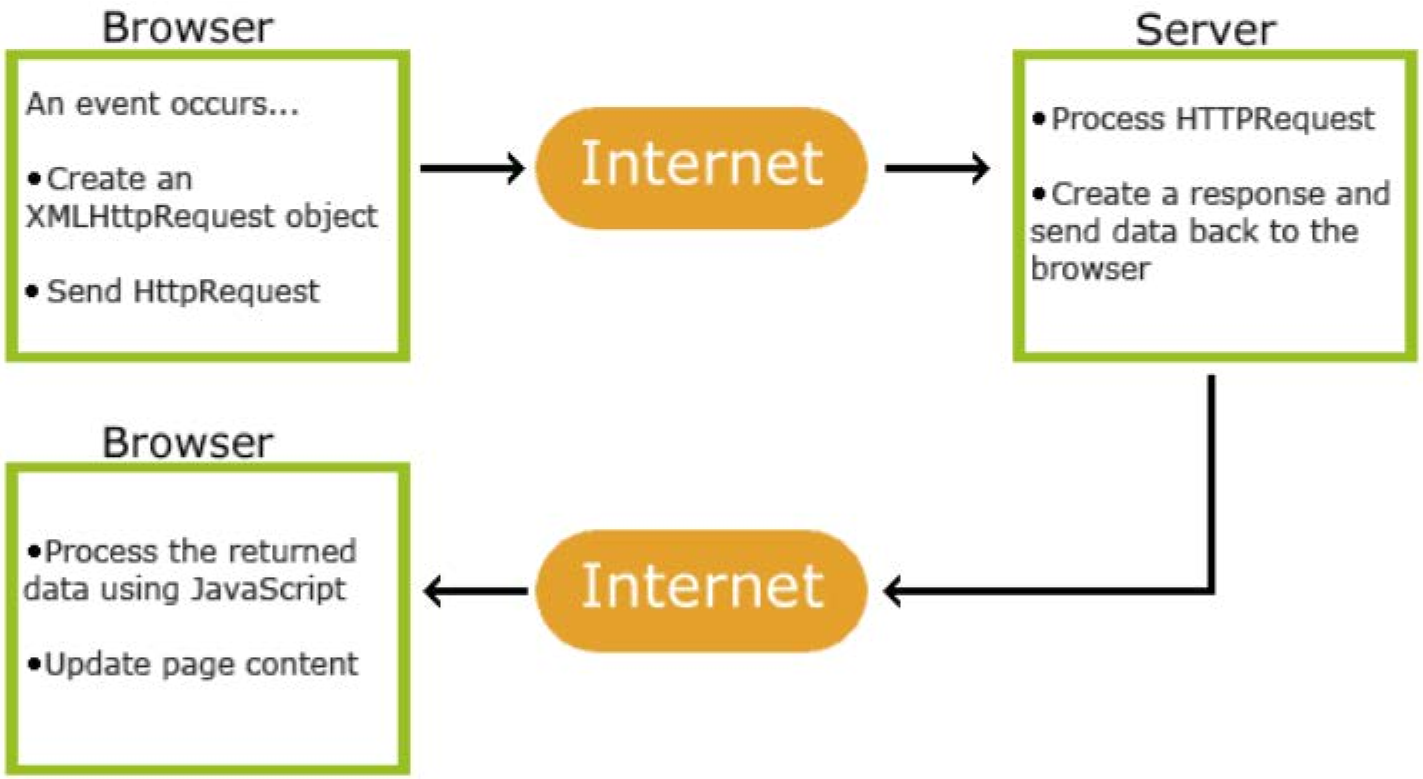
- Land Rover
- 17
- 15

AJAX

AJAX

- JAX is about updating parts of a web page, without reloading the whole page.
- AJAX = Asynchronous JavaScript and XML.
- AJAX allows web pages to be updated asynchronously by exchanging small amounts of data with the server behind the scenes. This means that it is possible to update parts of a web page, without reloading the whole page.
- Examples of applications using AJAX: Google Maps, Gmail, Youtube, and Facebook tabs.

HOW AJAX WORKS


Browser
An event occurs...
•Create an XMLHttpRequest object
•Send HttpRequest
Internet
Server
•Process HTTPRequest
•Create a response and send data back to the browser
Browser
•Process the returned data using JavaScript
•Update page content
Internet

THE END