**Advanced Techniques**

**Multidimensional arrays**

Sorting

$a = [

[

‘key1’ => 940,

‘key2’ => ‘blah’

],

[

‘key1’ => 23,

‘key2’ => ‘this’

],

[

‘key1’ => 894,

‘key2’ => ‘that’

]

];

This is a simple two-dimensional array (an array whose elements are also arrays), that you might need to sort using *key1* (numeric), or *key2* (alphabetical).

To sort a multidimensional array, you define your own *sort* function and then tell PHP to use that function by invoking the built in *usort90, uasort()* or *uksort()* function.

The function you define must take 2 parameters and return a value indicating which parameter should be first in the sorted list. A negative or false value means that the first parameter should be listed before the second. A positive or true value means that the second parameter should come first. A value of 0 indicates the parameters have the same value.

Function asc\_number\_sort($x, $y) {

If($x[‘key1’] > $y[‘key1’]) {

Return true;

}else if($x[‘key1’ < $y[‘key1’]]) {

Return false;

}else{

Return 0;

}

}

Usort($a, ‘asc\_number\_sort’);

The *usort()* function sorts by values but does not maintain the keys (for outer-most array). When you use the *uasort()*, the keys will be maintained. When you use *uksort()*, the sort is based on the keys.

Function string\_sort($x, $y) {

Return strcasecmp($x[‘key2’], $y[‘key2’]);

}

Usort($a, ‘string\_sort’);

\*More examples p5-p7\*

**List()**

List() assigns variables as if they were an array.

$my\_array = [‘dog’, ‘cat’, ‘horse’];

List($a, $b, $c) = $my\_array;

Echo “I have several animals, a $a, a $b and a $c”;

**Typehinting**

Typehinting is the act of indicating what type a variable needs to be.

Function foo(array $input) {}

If the function is called without providing the correct type (array in the example above), an error will be triggered.

**Advanced function definitions**

Recursive function

A recursive function is the act of a function calling itself:

Function foo() {

//some code

Foo();

//possible other code

}

The end result of a recursion is that the functions code is executed repeatedly, as if called from within a loop.

Recursive functions are necessary when you have a process that would be followed to an unknown depth. For example, a script that searches through a directory may have to search through any number of subdirectories:

Function list\_dir($start) {

$contents = scandir($start);

Foreach($contents as $item) {

If(is\_dir(“$start/$item”) && (substr($item, 0, 1) !== ‘.’)) {

//use item

List\_dir(“$start/$item”);

}else{

//use item

}

}

}

List\_dir(‘.’);

**Static variables**

When working with recursion, or any script in which the same function may be called multiple times, you might want to consider using the *static* statement. *Static* forces the function to remember the value of a variable from function call to function call, without using global variables.

Function howMany() {

Static $count = 1;

//code here

Return $count;

$count ++;

}

**Anonymous functions (AKA lambdas)**

i.e. a function without a name:

$var = function($b) { //code }

The function needs to be assigned to a variable. You would invoke it like:

$var(‘hi’);

Anonymous functions are best used on PHP functions which take a function as an argument and don’t need to be re-used.

Array\_map(function($val) {

//code

}, $names\_array);

The positive to anonymous functions is that PHP will only need to maintain the anonymous functions definition while the function is being directly used.

**The heredoc syntax**

Heredoc is an alternative way for encapsulating strings. It is used much less often than the standard double or single quotes but it fulfils the same role.

<<<EOT //starts with <<< followed by alphanumeric combination (any)

//code here

EOT; //must end with same delimiter which was used at the start

Nothing else should go after the identifier, not even a space!

$string = <<<EOD

Hello world

EOD;

This is a good alternative when writing multiple lines of HTML which will include quotation marks etc…

$link = ‘google’;

Echo <<<HTML

<div id=”container”>

<a href=<http://google.com>>$link</a>

</div>

HTML;